Journal of Visual Communication and Image Representation **10**, 379–397 (1999)
Article ID jvci.1999.0425, available online at http://www.idealibrary.com on IDEAL®

On Reversible Skeletonization Using Anchor-Points from Distance Transforms

Stina Svensson and Gunilla Borgefors

Centre for Image Analysis, Swedish University of Agricultural Sciences, Lägerhyddvägen 17, SE-752 37, Uppsala, Sweden
E-mail:stina@cb.uu.se, gunilla@cb.uu.se

and

Ingela Nyström

Centre for Image Analysis, Uppsala University, Lägerhyddvägen 17, SE-752 37, Uppsala, Sweden
E-mail: ingela@cb.uu.se

Received November 18, 1998; accepted May 25, 1999

In many applications thinning of objects is of great interest. We here present a skeletonization algorithm that is based on the idea of iteratively thinning the distance transform of an object layer by layer until either an anchor-point is reached or the connectivity breaks. Our algorithm is general in the sense that any metric and any connectivity can be used. Also, it is based on ideas that are not specific for 2D. The properties of the resulting skeletons are evaluated according to the "Lee–Lam–Suen properties." © 1999 Academic Press

Key Words: thinning; skeletonization; centers of maximal disks; digital topology; distance transforms; Hilditch crossing number; Rutovitz crossing number.

1. INTRODUCTION

Thinning of objects is a very helpful tool in many image analysis applications, e.g., character recognition, inspection of printed circuit boards, analysis of mineral and vegetable fibers, analysis of chromosome shapes, analysis of white blood cells, examination of soil cracking patterns, quantitative metallography, and classification of fingerprints. It is generally useful for shape analysis, object matching, and quality control. Since the 1960s, many papers have been published on this subject and among those many good skeletonization algorithms can be found. Two main classes of skeletonization methods have developed: iterative thinning and distance transform analysis. We present an approach using the best of both methods.

Existing 2D skeletonization algorithms are usually not straightforward to extend to higher dimensions. Existing volume (3D) skeletonization algorithms are usually not rotation

1047-3203/99 $30.00

stable, quite complex to compute, or applicable only to a limited class of objects. The concept of skeletonization has also been extended to 4D data, e.g., Jonker and Vermeij [1]. The motivation for our skeletonization algorithm was the aim of developing a general skeletonization algorithm. The goal is an algorithm producing skeletons stable under rotation, eventually working in any dimension. The new 2D skeletonization algorithm that resulted has its own unique merits. It is simple to compute and can be based on any metric and any connectivity. This means it can be adapted to the general shape of objects in the application and, using the digital Euclidean metric, be as rotation independent as digitization allows. This paper presents this new skeletonization algorithm and its result.

We start with giving some important definitions in Section 2. In Section 3, the background of different skeletonization algorithms and the motivation for one more 2D skeletonization algorithm is given. Our method is described in Section 4, and in Section 5 we evaluate the performance of our skeletons according to some general properties listed in the evaluation paper by Lee, Lam, and Suen [2]. We show some image examples in Section 6. In Section 7, postprocessing of the skeleton is discussed and in Section 8 the performance of the skeleton in the presence of noise is evaluated.

2. DEFINITIONS

The central pixel in a 3×3 neighborhood has two types of neighbors: four pixels sharing an edge and four pixels sharing a point with the central pixel. These originate two different connectivities: 4-connectivity defined by the edge-neighbors and 8-connectivity defined by the edge- and point-neighbors. In a binary two-dimensional image an *object* is defined as an 8-connected set of pixels and the *background* as a 4-connected set of pixels. Both object and background can consist of several components.

Each object pixel can be labeled with the *distance* to the closest background pixel. The metric to choose when computing the distance depends on the application. The two simplest metrics that are used for two-dimensional images are the "path-generated" distances D^4, or city block, and D^8, or chessboard (Rosenfeld and Pfaltz [3]), where the distance is the number of steps in the minimal path between pixels with 4- and 8-connectedness, respectively. In other words, the distance between a point $\mathbf{x} = (x_1, x_2) \in \mathbf{Z}^2$ and a point $\mathbf{y} = (y_1, y_2) \in \mathbf{Z}^2$ is

$$D^4(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{2} |y_i - x_i|,$$

$$D^8(\mathbf{x}, \mathbf{y}) = \max_{i=1,2} |y_i - x_i|.$$

In some cases, a metric is needed that is as similar as possible to the metric usually thought of when measuring distances in the continuous world, i.e., the Euclidean distance (Danielsson [4]). It has the advantage of being as rotation invariant as digitization allows. The ideal Euclidean distance would be that the distance between a point $\mathbf{x} = (x_1, x_2) \in \mathbf{Z}^2$ and a point $\mathbf{y} = (y_1, y_2) \in \mathbf{Z}^2$ is

$$D^E(\mathbf{x}, \mathbf{y}) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}.$$

To compute the distances in an image a *distance transformation* (DT) is used. The DTs for D^4 and D^8, DT^4 and DT^8, respectively, are described by Rosenfeld and Pfaltz in [5].

However, if the Euclidean distance is needed these are bad approximations. Since the Euclidean distance is computationally heavy, at least when extending to higher dimensions, and leads to complex rules in most applications, skeletonization not the least (Borgefors [6]), the so-called weighted distance transforms (WDT) (Borgefors [7]) are often used. Here, different weights are used to measure unit steps in various directions. For a 3×3 neighborhood we have WDT on the form a–b, where a is the distance to an edge-neighbor and b the distance to a point-neighbor. Different integer weights are found optimal depending on the optimization condition used. Common weights are 2–3, 3–4, and 5–7. For a 5×5 neighborhood we have WDT on the form a–b–c, where a and b are as above and c is the distance to a "knight"-neighbor. Optimal integer weights here are 5–7–11, showed by Borgefors in [7]. On the digital Euclidean DT (EDT), see Danielsson [4].

To compute the distance transform of an image, using D^4, D^8, a–b, or a–b–c, a simple two-pass sequential algorithm can be used (Rosenfeld and Pfaltz [5], Borgefors [7]). The algorithm performs a propagation with increasing distances in a forward and a backward scan. Each pixel is assigned the minimum of the value of the current pixel and the sums of the already visited neighbors and their corresponding local distances to the current pixel. The size of the neighborhood depends on the chosen metric. For the digital Euclidean metric this is a little more complicated. A 2D vector giving the x and y distances to the nearest background pixel must be stored and updated for each pixel. In this paper we will use Danielsson's, [4], 3×3 neighborhood Euclidean distance transform that uses four passes over the image.

Starting from a number of seed pixels with distance values, the *reverse distance transform* can be computed. A simple two-pass sequential algorithm for this, analogue to the one mentioned above, performs a propagation with decreasing distances in a forward and a backward scan. Each pixel is assigned the maximum of the value of the current pixel and the differences between the already visited neighbors and their corresponding local distances to the current pixel. The algorithm works when using D^4, D^8, a–b, or a–b–c metrics. For the Euclidean metric the reverse DT is much more complicated; see Borgefors and Nyström [8].

The distance label of any pixel in the distance transform can be interpreted as the radius of a disk, in the chosen metric, centered on the pixel. Per definition, this disk is included in the object. An object pixel is called *center of a maximal disk* (CMD) if the associated disk is not completely covered by any other single disk. The union of the maximal disks, which might be computed by applying the reverse DT with the set of CMDs as seed values, coincides with the whole object.

For path-generated DTs, the CMDs are equivalent to the local maxima (Rosenfeld and Pfaltz [3]). When the WDT is used, the weights have to be taken into account to detect the CMDs. The definition of CMDs becomes, from Borgefors [9, Definitions 3 and 4], the following.

DEFINITION 1. A pixel p in an a–b (a–b–c) WDT valued v_c, where $v_c \geq V$, is a CMD if $v_e < v_c + a$ for all edge-neighbors v_e and $v_p < v_c + b$ for all point-neighbors v_p (and $v_k < v_c + c$ for all knight-neighbors v_k), where V is a unique value associated with a and b (a, b, and c). For $v_c < V$ special rules must be used.

When detecting CMDs using the 3–4 DT, the only special cases to consider are pixels with distance values 3 and 6. If these are replaced by values 1 and 5, respectively, the general criteria can be used (Arcelli and Sanniti di Baja [10]). When detecting CMDs using the 5–7–11 DT, $V = 61$ and, thus, there are many special cases. For more information about

detecting CMDs for WDT and how to compute the special cases, see Arcelli and Sanniti di Baja [10] and Borgefors [9]. Details about how to detect the CMDs for the EDT are found in Borgefors *et al.* [11]. This has to be done by look-up tables.

Per definition, no disk associated with a CMD is covered by a single disk associated with any other CMD, but it can happen that a maximal disk is completely covered by the union of several other maximal disks; i.e., the set of CMDs contains more pixels than necessary. Thus, it is possible to find a subset, the *reduced set of centers of maximal disks* (RCMDs), of the CMDs that fully describe the object. A way to find the "optimal" set of CMDs would be to test all possible subsets of the set of CMDs, save those which cover the object, and then select the subset with the fewest disks. This is, however, very time consuming. There are other methods to find a considerably reduced set of centers of maximal disks, even though this may not be the optimal set (Borgefors and Nyström [8], Nilsson and Danielsson [12]). We will use the method described in [8]. The idea is to have a list with all CMDs, sorted by increasing distance value, and their positions. A temporary image is used for storing the number of disks covering each pixel in the original image. Traverse the list of CMDs, in increasing radius order, and inspect in the temporary image to check whether the corresponding disk covers pixels that all have values greater than one. If that is the case, the CMD can be removed from the list.

A *layer k*, in a DT of an object, is defined as the set of pixels reached in k steps from the background. For DT^4 and DT^8, this means that the layers are identified as 8- and 4-connected sets, respectively, consisting of pixels labeled with the same distance values. For a WDT a pixel with distance value p belongs to the kth layer (Arcelli and Sanniti di Baja [13]), if

$$a \cdot (k-1) < p \leq a \cdot k,$$

where a is the distance to an edge-neighbor. For the EDT (Borgefors *et al.* [14]), a pixel with distance value p belongs to the kth layer if

$$(k-1)^2 < p \leq k^2.$$

For *topology preservation* we have the following definition of Kong and Rosenfeld [15].

DEFINITION 2. An object pixel p in a two-dimensional digital image is called a simple point if the deletion preserves the topology in the sense that both the number of object components and the number of background components stay the same when p is removed.

Many different criteria to preserve the topology while thinning are described in the literature. Common criteria are either to use a large number of masks or to use any of the *crossing numbers.* The crossing numbers are computed by investigating the neighborhood of a pixel p. The eight neighbors of p are denoted n_i, where $i = 1, \ldots, 8$, as shown in Fig. 1. There are (at least) two different definitions of a crossing number. The first was introduced

n_4	n_3	n_2
n_5	p	n_1
n_6	n_7	n_8

FIG. 1. The neighbors of a pixel p.

by Rutovitz in [16] and is defined as

$$X_R(p) = \sum_{i=1}^{8} |n_{i+1} - n_i|,$$

where $n_9 = n_1 \cdot X_R$ is equal to two times the number of 4-connected components in the neighborhood of p; i.e., if $X_R = 2$ the 4-connectivity is not affected when p is removed. Hilditch gives, in [17], a different definition of a crossing number. Here, the 8-connectivity is taken into account by the following,

$$X_H(p) = \sum_{i=1}^{4} b_i,$$

where

$$b_i = \begin{cases} 1 & \text{if } n_{2i-1} = 0 \text{ and } (n_{2i} = 1 \text{ or } n_{2i+1} = 1) \\ 0 & \text{otherwise} \end{cases}.$$

X_H is equal to the number of 8-connected components of the object in the neighborhood of p, except when all four edge-neighbors of p are object pixels. Then $X_H = 0$. If $X_H = 1$, the removal of p does not break the 8-connectedness of the object. Equally, this gives us a measure of the number of 4-connected components in the background that has the center pixel as an edge-neighbor; two components break the connectedness of the background, one retains it. Both crossing numbers give the same result for an isolated pixel as for a pixel with only object pixels as neighbors; i.e., $X_R = X_H = 0$.

The Hilditch crossing number has later been defined in an equivalent, but more readily computable form in Yokoi *et al.* [18], called the 8-connectivity number. In the same article the 4-connectivity number, equivalent to X_R, also is introduced.

3. BACKGROUND

Numerous good skeletonization algorithms are described in the literature. In Lam, Lee, and Suen [19] a survey of thinning, or skeletonization, algorithms is presented. When that paper was written, in August 1990, about 300 articles had been published on various aspects of the subject. In the article many different thinning algorithms are discussed. The authors have, deliberately, left out skeletonization algorithms based on DTs. The ones discussed, i.e., nondistance-based algorithms, have the advantage that no DT has to be computed for the object and, because of that, also the serious drawback that the skeleton is not reversible (in most cases).

Most skeletonization algorithms found in the literature use an iterative thinning process and some criteria describing when a pixel can be deleted without breaking the connectivity. One approach classifies some pixels as *anchor-points*, i.e., nonremovable, before the thinning starts. Using the CMDs as anchor-points during the thinning process guarantees reversible skeletons. In Vincent [20] the full set of CMDs are used as a first step in the skeletonization algorithm. Since this set is not topologically equivalent to the original object, connecting arcs are needed for topology preservation. These are detected by iteratively removing pixels while not breaking the connectivity. For the removal a set of masks is used. All possible configurations of a neighborhood are stored in a look-up table with a marker indicating whether the pixel in the current neighborhood can be removed or not. In [20] only

the hexagonal grid and the "honeycomb" DT (equivalent to the DT^4) is considered. In this case the number of configurations in the neighbourhood is 2^6, i.e., the look-up table must contain 64 elements. Queues are used to obtain a fast implementation. The method is easy to implement for the square grid. However, a considerably larger look-up table is then needed.

Also, in Ragnemalm [21] a skeleton using the CMDs, or the so-called α-skeleton, as anchor-points is presented. The α-skeleton was introduced by Kruse in [22]. Ragnemalm uses CMDs from the Euclidean metric. The object is iteratively thinned until an anchor-point is reached or the connectivity breaks. The connectivity criterion uses 20 different masks. If a thin 8-connected skeleton is wanted a pre- or postprocessing step has to be added, using another 16 masks. The algorithm produces nice skeletons that are rotation invariant, up to sampling effects, have high reconstructibility, and are connectivity preserving.

A slightly different, noniterative, approach for the 3–4 DT is described by Sanniti di Baja in [23]. CMDs are also detected here and used as anchor-points. *Saddle pixels*, defined as a pixel that has at least one 8-connected component of pixels labeled more than the pixel itself and more than one 4-connected component labeled less than the pixel itself in its neighborhood, are also detected. CMDs and saddle pixels are connected by paths. When a pixel is identified as a CMD or saddle pixel, the 3×3 neighborhood is inspected. It includes at most two 8-connected components labeled more than the pixel itself. For each component, the neighboring pixel for which the gradient has the maximal value is marked as a skeletal pixel. This process goes on until no pixel with positive gradient can be found. After that hole filling, final thinning, pruning, and beautification to some threshold is performed. This method is later described also for DT^4, DT^8, and 5–7–11 by Sanniti di Baja and Thiel in [24]. The Euclidean case is presented by Arcelli and Sanniti di Baja in [25]. Also, these algorithms have high reconstructibility, are connectivity preserving, and are, if based on a rotation invariant metric, rotation invariant. The skeleton can be computed by a fixed number of scans and no iterations are needed.

The above described, for 2D, well-working algorithms would be very hard to extend to higher dimensions. If, e.g., a number of masks are needed for the 2D case (Ragnemalm [21], Vincent [20]), several times more masks than for the 2D case would be needed for the 3D case, since the topology is much more complex. For the Sanniti di Baja method in [23], we have the problem of defining and detecting 3D saddle voxels, which in 3D probably needs a larger neighborhood than $3 \times 3 \times 3$, and most of all detecting the connecting surfaces; i.e., the CMDs and saddle points must be linked not by 3D paths but by 3D surfaces. A problem present in any dimension is that using CMDs as anchor-points (often) results in a more than one-pixel-thick skeleton. Hence, a final thinning is needed to obtain a thin skeleton. By using the RCMDs instead, the original skeleton will (often) be much thinner. Final thinning may still be needed but the reversibility is better starting from a skeleton with RCMDs as anchor-points than one with CMDs, as seen in Section 7. With the approaches by Vincent, Ragnemalm, and Sanniti di Baja as inspirations, we have tried to compute similar skeletons without the drawbacks mentioned in this paragraph.

4. SKELETONIZATION USING ANCHOR-POINTS

The algorithm will first be introduced briefly and then each step will be explained. The algorithm can be summarized as

1. Compute the DT of the object.

2. Detect the set of CMDs.
3. Reduce the set of CMDs to the RCMDs. Those will be our anchor-points.
4. Iteratively thin the object, distance layer by distance layer, using the Hilditch crossing number for topology preservation and keeping all anchor-points.

In a binary image a pixel is either object (1) or background (0). The first step is to compute the DT of the object. For our algorithm it is possible to use any metric, which is a great advantage, since the metric can be chosen depending on the shape of the object and the computational power available. From the DT, the set of CMDs is detected and thereafter the RCMDs, which will be our set of anchor-points. In this way we can be certain that the skeleton always will be fully reversible by the reverse DT, as described in Section 2.

After this we thin the object, distance layer by distance layer, until an anchor-point is reached or a pixel is found to be multiple, i.e., nonsimple. The distance layers are used as guides for the iteration process to get a nonbiased removal of simple pixels and thus keeping the skeleton centrally located in the object also in areas without anchor-points. To determine whether a pixel p is multiple or not, we use the Hilditch crossing number [17]. As mentioned earlier, the object is 8-connected and the background is 4-connected. By using the Hilditch crossing number we know that the resulting skeleton will be 8-connected, except where the set of anchor-points is thicker. The algorithm used for computing $M = 2X_H(p)$, i.e., twice the Hilditch crossing number, can be summarized by the following pseudo-code, here $n_9 = n_1$:

```
j:=1
M:=0
while (j<9)
   if (j=odd and n_j=1)
     then
       M:=M+|n_{j+2}−n_j|
       j:=j+2
     else
       M:=M+|n_{j+1}−n_j|
       j:=j+1
   endif
endwhile
```

If $M = 2$, the pixel is simple and can be removed without changing the topology. If $M > 2$ a multiple pixel is found. If $M = 0$ removing the pixel would create a hole. The algorithm works in a 3×3 neighborhood as described by the examples in Fig. 2.

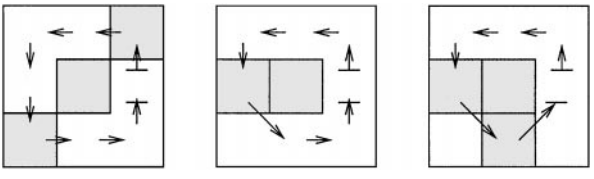


FIG. 2. To the left, an example of a multiple point ($M = 4$), and in the middle and to the right, examples of simple points ($M = 2$).

To get a topologically correct removal of simple pixels, the pixels belonging to a certain distance layer k must be traversed in the following way:

1. Mark all pixels belonging to distance layer k that has an edge-neighbor in the background, which are simple and are not anchor-points.
2. Sequentially remove all marked pixels that are still simple.

The reason for the second step is that a simple pixel can become multiple when one of its neighbors is removed; thus, the removal cannot be done simultaneously but must be done pixel by pixel. The marking, on the other hand, should be done for all pixels simultaneously. The Hilditch crossing number has to be computed in both steps since the neighborhood of each pixel may differ between step 1 and step 2. In step 2 already visited neighbors in the same layer may have been removed. The two steps are repeated for the same distance layer, until every pixel belonging to distance layer k is either removed or a skeletal pixel; i.e., no $M = 0$ are detected. $M = 0$ is equivalent to p not having an edge-neighbor in the background. Pixels in the same distance layer are traversed at most twice. That is because the set of pixels in a distance layer can be at most 4-connected (in all metrics considered here, it is not true for all possible metrics). Thus, if a pixel does not have an edge-neighbor in the background, it has at least one edge-neighbor in the same distance layer, which has an edge-neighbor in the background. The sets of pixels belonging to the same distance layer can be locally 4-connected in any metric, but this is most common in D^8, where it is the normal situation.

Pixels in the current distance layer that are neither removed nor marked as skeletal pixels during the two iterations must be contained in the skeleton, to avoid creating spurious holes. This situation is not (very) common, but appears when we have so-called *Arcelli sets* [26], i.e., when multiple pixels confine pixels in a higher distance layer. This may appear for any metric. In Fig. 3 an example of an Arcelli set is shown. If the D^8 metric is used all "border pixels" have the distance value 1, i.e., the pixels belong to distance layer 1, and the center pixel has distance value 2, i.e., the pixel belongs to distance layer 2. All border pixels are anchor-points or multiple and cannot be removed. The skeleton will be equivalent to the whole object, which is as it should be as no pixel can be removed without changing the topology of the object.

In some applications the skeleton of the background, i.e., the exoskeleton, is of interest. Still, we want the object to be 8-connected and the background 4-connected. This implies that also the skeleton should be 4-connected. Using the Rutovitz crossing number instead of the Hilditch crossing number when determining whether a pixel is multiple or simple will result in a 4-connected skeleton. The rest of the algorithm is the same.

Since our set of anchor-points is not necessarily one-pixel thick, it may happen that our algorithm produces a skeleton that is not one-pixel thick. If a one-pixel-thick skeleton is needed, a final thinning method, e.g., the one described by Sanniti di Baja in [23], can be

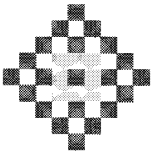


FIG. 3. A situation where $M = 0$ for several pixels in two subsequent iterations. The set of CMDs, which is the same as the RCMDs, is shown in dark gray. The D^8 metric is used.

used. Note that this final thinning will make the skeleton nonreversible, even if we know that only (a few) border pixels of the original object are lost by removing anchor-points in "thick" parts of the skeleton. Final thinning is discussed further in Section 7.

For many applications the skeleton needs to be pruned to remove nonsignificant branches. Pruning methods are described by, e.g., Sanniti di Baja and Thiel in [27] and also by Sanniti di Baja in [23]. These methods take skeleton branches and investigate the effects of branch removal by comparing the recovered regions. Note that pruning will make the skeleton nonreversible.

5. LEE–LAM–SUEN PROPERTIES

In a follow-up article to the thinning survey by Lam, Lee, and Suen [19], a systematic evaluation of skeletonization algorithms is made by Lee, Lam, and Suen [2]. The authors have found that there is a general agreement about properties that should be retained by a skeleton. These are summarized by 10 properties. The authors have then chosen 20 different skeletonization algorithms, most of them widely quoted in the literature. These are then compared with each other according to the 10 different properties. We will test our new skeletonization algorithm according to the properties to ensure that it measures among the best.

Property 1. The skeleton should accurately reflect the shape of the original pattern.

The authors point out that the interpretation of this statement is not unique. It can either mean that the skeleton should visually resemble the original shape or that the object is reconstructible. In any case, the skeleton should be topologically equivalent to the original object.

Our skeleton reflects the shape as well as possible depending on the choice of metric; see Section 6. The object is always reconstructible, since every member of the RCMDs is contained in the skeleton. Thereby, Property 1 is fulfilled.

Property 2. The connectedness of both the pattern and the background should be preserved.

In our case, the connectedness of both the pattern and the background is preserved, since we only remove simple points.

Property 3. Points should be stripped off symmetrically.

This means that the skeleton should be located centrally in the object, e.g., at the medial axis.

In our skeletonization algorithm, the anchor-points are the RCMDs. The RCMDs are a subset of the "medial" line and are by definition centrally located, except for sampling effects. Also, thinning is performed distance layer by distance layer, so the points are removed symmetrically. Thus, the skeleton is located centrally as measured by the chosen metric.

Property 4. Rectangles whose length and width are both greater than 1 should not remain unchanged.

If we have a rectangle exactly two pixels wide our skeleton is equivalent to the whole object. This is because all pixels will be CMDs and the set of CMDs cannot be reduced. Our skeleton reduces any rectangle with length or width greater than two.

Note that Property 4 (and 7) precludes complete reversibility and is thus incompatible with one of the interpretations of Property 1.

Property 5. Results should be isotropic, i.e., invariance with respect to rotations.

If the metric chosen is rotation invariant our skeleton also will be rotation invariant, up to sampling effects. The best metric to choose to fulfill this property is the Euclidean one. The 3–4 metric and the 5–7–11 metric will perform nearly as well as the Euclidean metric, and the 3–4 metric is less time consuming (especially when computing the RCMDs).

Property 6. End-points should be retained.

Property 6 is to avoid erosion of significant protrusions.

The endpoints in our skeleton are included in the set of anchor-points and will thereby be retained. However, if a "protrusion" happens to be locally a sector of a digital disk in the metric used, then the endpoint will be the center of that disk and the protrusion will be visibly "lost" by the skeleton. It will, however, be included in the reconstruction from the skeleton.

Property 7. The resulting skeleton should have single pixel width.

Since the RCMDs are not always single-pixel width it may happen that our skeleton does not have single-pixel width everywhere. This is necessary, if we want to have a reversible skeleton. This always happens when the object has either a branch or a bridge that has even pixel width, as described under Proposition 4. However, if the skeleton is thinned to single-pixel width, the skeleton will still be almost reversible. Only border pixels of the original shape can be lost. Final thinning of our skeleton to single-pixel width is discussed in Section 7.

Property 8. The skeletonization process should be immune to noise.

With this formulation Lee, Lam, and Suen mean that no spurious branches should be contained in the skeleton and that "the generation of noise spurs should be minimized." This property is the only one were we have some reservations. The reason is that the definition of "noise" is so difficult and varying. In Section 8 noise is further discussed, and our algorithm is evaluated according to a realistic noise model used by Jaisimha, Haralick, and Dori in [28] (where different thinning algorithms in the presence of noise are evaluated). The results show that our skeleton can be made reasonably robust to this type of noise.

Property 9. The resulting skeleton should be obtained in a short time.

Our algorithm is fast for the fastest DTs, i.e., DT^4, DT^8, and 3–4 DT. The EDT and especially 5–7–11 DT require some more time. In Table 1 the time needed for computing

TABLE 1
The Approximate CPU Seconds Needed for Each Individual Process When Calculating the Skeleton of a "Residential District" with 16 "Houses" in a 256 × 256 Image

	CPU seconds				
Metric	DT	CMDs	RCMDs	Thinning	Total
D^4	0.02	0.01	0.07	0.35	0.45
D^8	0.02	0.01	0.03	0.35	0.41
3–4	0.02	0.01	0.10	0.38	0.51
5–7–11	0.03	0.01	1.32	0.38	1.74
Euclidean	0.06	0.01	0.35	0.35	0.76

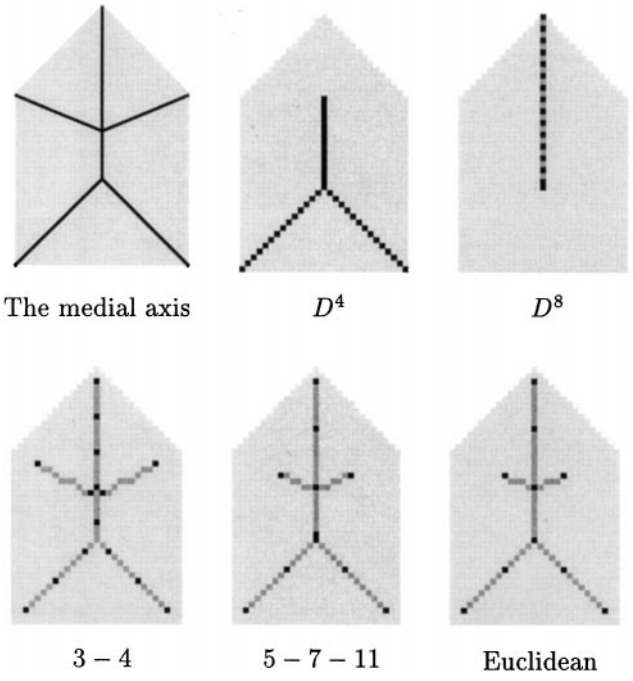


FIG. 4. Medial axis (continuous) and skeleton for the tested metrics. The sets of anchor-points are shown in black.

the skeleton when run on a DEC-Alpha (a standard UNIX workstation) is shown. The time is specified as the time required for computing the DT, detecting the CMDs, reducing the set of CMDs, and performing the thinning. The image used is a 256×256 image, with 17,056 object pixels, which pictures a "residential district," consisting of 16 of the "houses" from Fig. 4. The algorithm has not been optimized, but the relative ranking of DTs should be correct. The reason why the slowest metric to use is 5–7–11 is that a 5×5 neighborhood has to be used, instead of, as for the other metrics (including the Euclidean metric), only a 3×3 neighborhood.

Compared to many other distance-transform-based algorithms our algorithm often is more computationally complex due to its iterative nature. However, with the existing computers it is fast enough for most applications.

Property 10. The skeletonization algorithm should be easily implemented by hardware techniques such as parallel processing for real time applications.

Most parts of our algorithm could easily be implemented by SIMD-parallel processing. The main exception is the removal of marked pixels during thinning. Also the detection of the RCMDs requires some sequentiality. However, with today's fast sequential computers the images have to be quite large and contain quite thick objects to take advantage of parallel processing.

6. IMAGE EXAMPLES

As mentioned earlier, our method works for any digital metric. In Fig. 4 resulting skeletons using a number of different metrics, D^4, D^8, 3–4, 5–7–11, and Euclidean, are shown. These

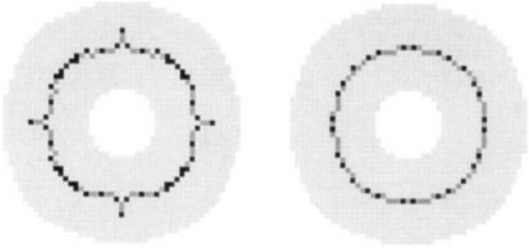


FIG. 5. Skeleton using DT^8 (left) and 5–7–11 (right). The sets of anchor-points are shown in black.

skeletons can be compared to the medial axis, [29], of the continuous version of the same object, also shown in Fig. 4. The object (light gray) is a simple "house" and consists of the union of a disk in D^8 (body) and half a disk in D^4 (roof). The skeleton (dark gray) and the anchor-points (black) are shown for each metric.

As shown in Fig. 4, different metrics produce different skeletons, all of them fully reversible. Disks in different metrics have different shapes, quadratic for D^8, diamond-shaped for D^4, octagonal for 3–4, and hexadecagonal for 5–7–11. If the object is expected to be quadratic or rectangular, D^8 is the best metric to choose, according to a compression point of view, but D^4 will give the best resemblance to the medial axis. If the object is expected to be more roundish the 3–4, the 5– 7–11, or the Euclidean metric should be chosen. An example is shown in Fig. 5, where the skeleton based on the D^8 and on the 5–7–11 metrics of a digital Euclidean ring are shown. Note that if the 3–4, the 5–7–11, or the Euclidean metric is chosen, the skeleton will be almost rotation invariant (3–4, 5–7–11) or rotation invariant (Euclidean).

In Fig. 6 the difference between using the set of CMDs and RCMDs as anchor-points, respectively, is shown for a "real" image of a horse.

7. FINAL THINNING

Other skeletonization algorithms use the full set of CMDs as anchor-points, e.g., Vincent [20] and Ragnemalm [21]. The skeleton will then have more than single-pixel width, since the set of CMDs often has more than single-pixel width. Often the skeleton is reduced to

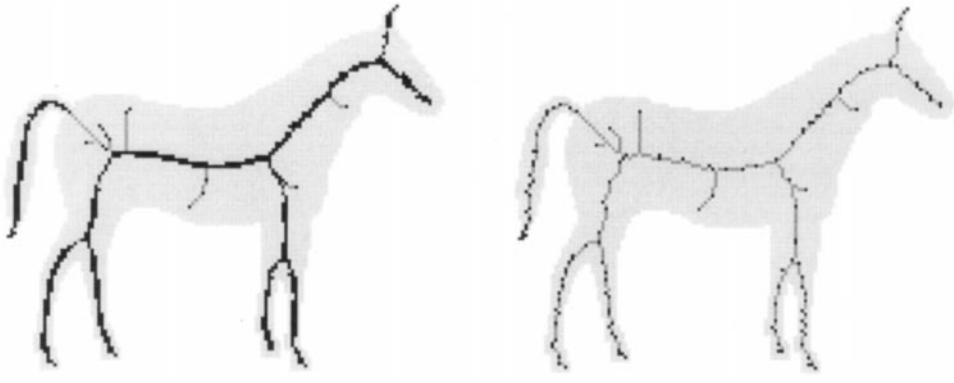


FIG. 6. Skeleton, without any postprocessing, using the 3–4 DT and the set of CMDs as anchor-points (left) and the RCMDs as anchor-points (right). The sets of anchor-points are shown in black.

single-pixel width in a (pre- or) postsprocessing step. This removal is random in the sense that CMDs essential for reversibility (the RCMDs) are as likely to be removed as those that are not essential. By identifying the RCMDs, this randomness is removed, and more of the shape is eventually recoverable. Thus, this is worth doing, even if many of the non-RCMDs' CMDs are still included in the skeleton. They are included since they are centrally located and essential for topology reasons (multiple).

To get single-pixel width in our skeletons a final thinning algorithm can be used, e.g., the one presented by Sanniti di Baja in [23]. According to that algorithm a skeleton pixel p can be removed from the skeleton if:

1. at least one odd-numbered neighbor of p is not a skeletal pixel. For the numbering of the neighbors, see Fig. 1.
2. at least one triple of neighbors n_i, n_{i+2}, n_{i+5} exists, i is odd, such that n_i and n_{i+2} are skeletal pixels, but n_{i+5} is not. Here $n_9 = n_1$, $n_{10} = n_2$, and so on.

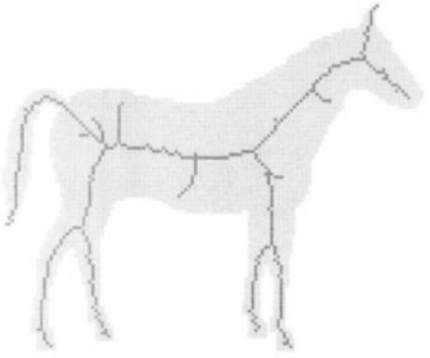
Criterion 1 is to avoid the creation of holes in the skeleton and criterion 2 for keeping the connectedness and avoiding shortening of branches.

There are (at least) three different ways of creating a unit wide skeleton using variants of our skeletonization algorithm. Two ways of creating the original skeleton are to use either the set of CMDs or RCMDs as anchor-points as in Fig. 6. The final thinning when the set of CMDs is used as anchor-points can then be performed in two different ways. In the first case the resulting skeleton is thinned once according to the final thinning algorithm mentioned above. The final skeleton and its recovery for the horse image in Fig. 6 are shown in Fig. 7 (top). In the second case the resulting skeleton is thinned not once but twice: first without removing any member of the RCMDs, using our skeletonization algorithm, and second to get single-pixel width, using the final thinning method mentioned above. As the set of CMDs in this case is thinned in a more controlled way, i.e., more of the necessary CMDs are kept, the reversibility performs better than in the first case; see Fig. 7 (middle). The third variant of the unit-wide skeleton is obtained by applying the final thinning when the RCMDs are used as anchor-points; see Fig. 7 (bottom). The reversibility of the skeleton performs best when the information of the RCMDs is used, but whether the RCMDs are used while producing the original skeleton or while doing the final thinning does not really matter. Note that these are general observations from testing on many shapes, but here only illustrated on the horse image using the 3–4 metric.

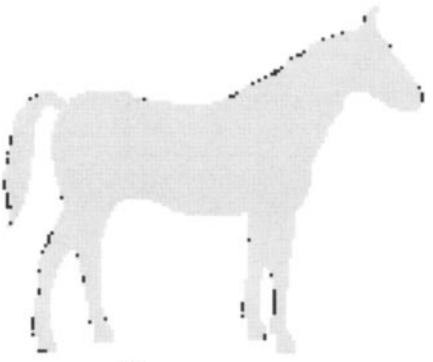
This section is meant to point out the difference between using the information in the full set of CMDs and using the information in the RCMDs. Note that no beautification, whatsoever, of the skeleton has been performed in any of the three methods. For some applications a less jagged skeleton is preferable and hence a final beautification has to be performed on the skeleton, e.g., the one described by Sanniti di Baja in [23].

8. SKELETONS OF NOISY OBJECTS

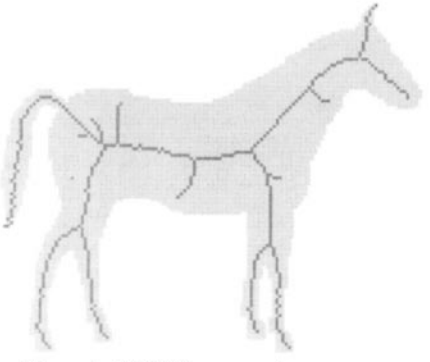
Among the Lee–Lam–Suen properties there is one, Property 8, that says that the skeletonization algorithm should be immune to noise. This is a very difficult property to evaluate, depending on the definition of noise. Different authors seem to mean anything from a few border pixels added or removed to quite large protrusions from a "recognizable" shape when talking about noise. To expect a reversible skeleton that satisfies Property 1 to disregard the protrusions in the latter case is not reasonable. If the protrusions are known to be noise,



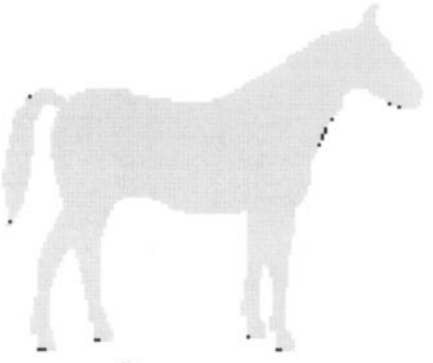
The set of CMDs as anchor-point.



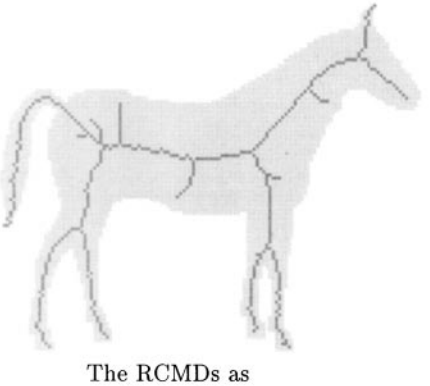
Recovery



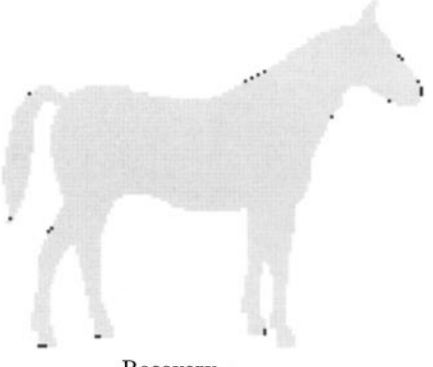
The set of CMDs as anchor-points. The RCMDs is used while doing the first step in the final thinning.



Recovery



The RCMDs as anchor-points.



Recovery

FIG. 7. Skeletons using the 3–4 DT that are postprocessed with a final thinning to unit width and the set of CMDs as anchor-points (top and middle) and the RCMDs as anchor-points (bottom) to the left. The middle skeleton is thinned twice, first without removing any member of the RCMDs and second to get single-pixel width. To the right is the recovery of the three skeletons. In black, pixels that are not recovered by applying the reverse DT.

they should be removed by, e.g., morphological operations in a preprocessing step. Nor is it reasonable to expect a skeleton to distinguish between significant parts of the object and noisy parts of the object if these parts are roughly of the same size, just because a human "recognizes" the shape and can do it.

In Jaisimha, Haralick, and Dori [28] an evaluation of quantitative performance of thinning algorithms in the presence of noise is presented. Here, noise is defined as degradation of documents due to digitization, photocopying, and printing. A perturbation model is used to simulate the noise, fully described by Kanungo, Haralick, and Philips in [30]. This is the type of minor border noise we consider reasonable to expect a skeleton to be invariant under.

In the perturbation model the probability of a background pixel becoming an object pixel, and vice versa, is dependent on the distance from the pixel to the object or the background. Let $P(o \mid D, \alpha, b)$ and $P(b \mid D, \alpha, b)$ be the probability of a background pixel, with distance D to the nearest object pixel, changing to an object pixel and remaining a background pixel, respectively, and let $P(b \mid D, \alpha, o)$ and $P(o \mid D, \alpha, o)$ be the corresponding probabilities for object pixels. After adding noise a morphological closing of the object is performed.

The functions used by Jaisimha, Haralick, and Dori in [28] are the following:

$$P(b \mid D, \alpha, b) = 1 - P(o \mid D, \alpha, b) = \alpha_0 e^{-\alpha D^2} + \eta \tag{1}$$

$$P(o \mid D, \beta, o) = 1 - P(b \mid D, \beta, o) = \beta_0 e^{-\beta D^2} + \eta. \tag{2}$$

Here, α and β are scale-factors for the amount of thinning and thickening, respectively, that the model will affect the original object.

In our algorithm, any noise that generates an anchor-point will also generate a branch starting from that anchor-point. This can be solved before the thinning by simply removing every pixel with value smaller than a certain (small) radius value from the DT to be thinned. The DT and the RCMDs are computed from the original object. This will not cause a shrinking of the nonnoisy parts of the object, as the remaining members of the RCMDs will include information of the original object size. The process will of course remove all branches of smaller radius, but if this is the known size of the noise, then none of these protrusions are significant. The same is also valid for thin connections. If a connection is thinner than the known size of noise the connection should be removed. This is the reason the pixels are removed from the DT and not only from the RCMDs. If we were only to remove small valued anchor-points then thin connections, e.g., a connection consisting of only one pixel, would be marked as multiple and hence not removed during thinning. Our method for avoiding spurious branches is similar to the method used by Vincent in [20]. The difference is that in [20] only small valued anchor-points are removed, thus keeping noisy connections in the skeleton.

Three examples are given in Fig. 8, where the (same) noise model was applied to the house image in Fig. 4. The probability functions (1) and (2) are used with $D = D^4$, $\alpha = 1$, $\alpha_0 = 1$, $\beta = 1$, $\beta_0 = 1$, and $\eta = 0$. Not only spurious protrusions, but also spurious holes should be avoided. As our skeleton is topologically correct, a spurious hole will generate a spurious loop, so all holes the size of noise should be and were filled before skeletonization. In the top row of Fig. 8 the skeletons obtained by using the algorithm described in Section 4, using the Hilditch crossing number and the 3–4 metric, are shown. In the middle row, all pixels valued one unit (here 3), including RCMDs, are removed before

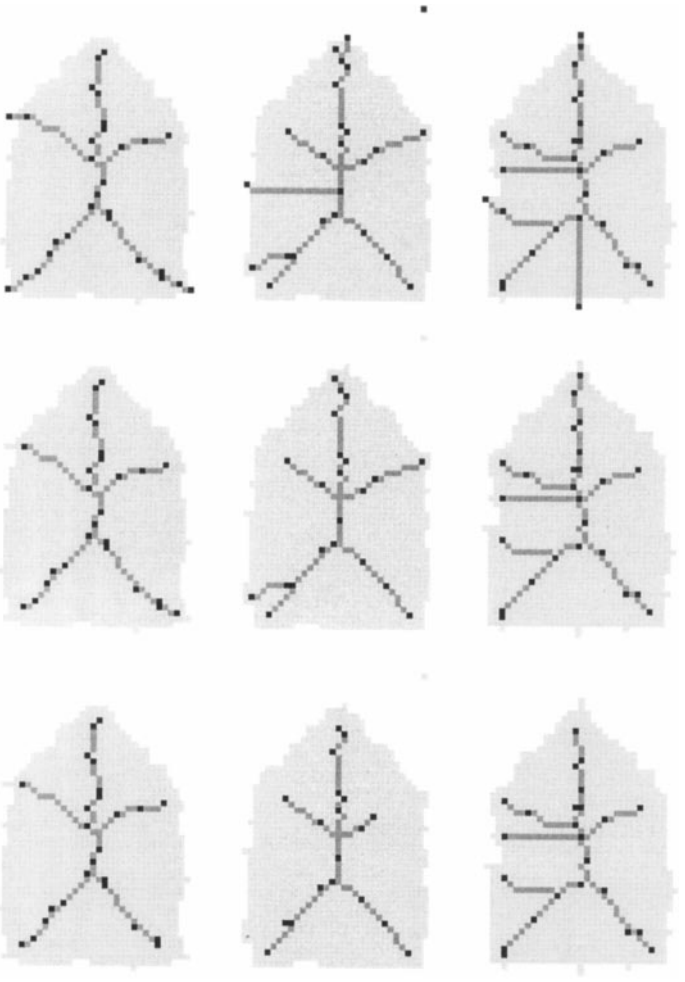


FIG. 8. Skeletons for different noisy images. The sets of anchor-points are shown in black. In the top row the skeleton of the noisy images. In the middle and the bottom row the skeleton when small-valued ($=3$ and $=6$, respectively) pixels are removed for the same images. The 3–4 metric is used and the original image is the same as in Fig. 4.

thinning. In this way, all noise with "radius" one unit is disregarded and spurious branches are removed. In the bottom row we show what will happen if all pixels valued two units (here 6), including RCMDs, are removed before thinning. It is of course possible to remove more pixels, which would remove even more spurious branches, but then significant branches would be shortened accordingly.

9. CONCLUSION

We have presented a general skeletonization algorithm. Our algorithm is based on the idea of iteratively thinning the distance transform of an object, layer by layer, until either an anchor-point is reached, i.e., a member of the reduced set of maximal disks, or the topology breaks. In this way, a general, fully reversible, at most two-pixel-thick skeleton is obtained without taking into account any special cases or using any set of masks. The thinning part of the skeletonization algorithm requires two passes over the image per iteration and the number of iterations is proportional to the thickness of the object to be skeletonized.

Our skeletonization algorithm is general in the sense that any metric can be used. Also, it is based on general ideas that not are specific for 2D, but can be extended to higher dimensions. In 3D there are ways to find the set of centers of maximal spheres and also to reduce that set (Borgefors and Nyström [8]). A way to count the connected components in the 3D neighborhood, is presented by Borgefors *et al.* in [31]. Future work will be to extend the algorithm described in this paper to higher dimensions.

ACKNOWLEDGMENTS

Dr. Gabriella Sanniti di Baja, Istituto di Cibernetica, Italian National Research Council, is gratefully acknowlegded for valuable comments. The authors also acknowledge the reviewers for their useful remarks and suggestions.

REFERENCES

1. P. P. Jonker and O. Vermeij, On skeletonization in 4D images, in *Proceedings of SSPR'96: Advances in Structural and Syntactical Pattern Recognition* (P. Perner, P. Wang, and A. Rosenfeld, Eds.), pp. 79–89, Springer-Verlag, Berlin/Heidelberg, 1996.
2. S. W. Lee, L. Lam, and C. Y. Suen, A systematic evaluation of skeletonization algorithms, in *Thinning Methodologies for Pattern Recognition* (S. W. Lee and P. S. P. Wang, Eds.), pp. 239–261, World Scientific, Singapore, 1994.
3. A. Rosenfeld and J. L. Pfaltz, Sequential operations in digital picture processing, *J. Assoc. Comput. Mach*. **13**, 1966, 471–494.
4. P.-E. Danielsson, Euclidean distance mapping, *Comput. Graphics Image Process*. **14**, 1980, 227–248.
5. A. Rosenfeld and J. L. Pfaltz, Distance functions on digital pictures, *Pattern Recogn*. **1**, 1968, 33–61.
6. G. Borgefors, Applications using distance transforms, in *Aspects of Visual Form Processing* (C. Arcelli, L. P. Cordella, and G. Sanniti di Baja, Eds.), pp. 83–108, World Scientific, Singapore, 1994.
7. G. Borgefors, Distance transformations in digital images, *Comput. Vision Graphics Image Process*. **34**, 1986 344–371.
8. G. Borgefors and I. Nyström, Efficient shape representation by minimizing the set of centres of maximal discs/spheres, *Pattern Recogn. Lett*. **18**, 1997, 465–472.
9. G. Borgefors, Centres of maximal discs in the 5-7-11 distance transform, in *Proceedings of Scandinavian Conference on Image Analysis (SCIA '93)* (K. A. Høgda, B. Braathen, and K. Heia, Eds.), pp. 105–111, Norwegian Society for Image Processing and Pattern Recognition, 1993.
10. C. Arcelli and G. Sanniti di Baja, Finding local maxima in a pseudo-Euclidean distance transform, *Comput. Vision Graphics Image Process*. **43**, 1988, 361–367.
11. G. Borgefors, I. Ragnemalm, and G. Sanniti di Baja, The Euclidean distance transform: Finding the local maxima and reconstructing the shape, in *Proceedings of Scandinavian Conference on Image Analysis (SCIA '91)* (P. Johansen and S. Olsen, Eds.), pp. 974–981, Pattern Recognition Society of Denmark, 1991.
12. F. Nilsson and P.-E. Danielsson, Finding the minimal set of maximum disks for binary objects, *Graphical Models Image Process*. **59**, 1997, 55–60.
13. C. Arcelli and G. Sanniti di Baja, Weighted distance transforms: A characterization, in *Image Analysis and Processing II. Proceedings of the Fourth International Conference on Image Analysis and Processing* (V. Cantoni, V. Di Gesù, and S. Levialdi, Eds.), pp. 205–211, Plenum Press, New York, 1988.
14. G. Borgefors, I. Ragnemalm, and G. Sanniti di Baja, Feature extraction on the Euclidean distance transform, in *Proceedings of the 6th ICIAP: Progress in Image Analysis and Processing II* (V. Cantoni, M. Ferretti, S. Levialdi, R. Negrini, and R. Stefanelli, Eds.), pp. 115–122, World Scientific, Singapore, 1991.
15. T. Y. Kong and A. Rosenfeld, Digital topology: Introduction and survey, *Comput. Vision Graphics Image Process*. **48**, 1989, 357–393.
16. D. Rutovitz, Pattern recognition, *Royal Statist. Soc*. **129**, 1966, 504–530.
17. C. J. Hilditch, Linear skeletons from square cupboards, *Mach. Intelligence* **4**, 1969, 403–420.
18. S. Yokoi, J.-I. Toriwaki, and T. Fukumura, An analysis of topological properties of digitized binary pictures using local features, *Comput. Graphics Image Process*. **4**, 1975, 63–73.
19. L. Lam, S. W. Lee, and C. Y. Suen, Thinning methodologies—a comprehensive survey, *IEEE Trans. Pattern Anal. Mach. Intell*. **14**, 1992, 869–885.

20. L. Vincent, Efficient computation of various types of skeletons, in *Proceedings of SPIE Medical Imaging V* (M. H. Loew, Eds.), pp. 297–311, 1991.
21. I. Ragnemalm, Rotation invariant skeletonization by thinning using anchorpoints, in *Scandinavian Conference on Image Analysis (SCIA '93)* (K. A. Høgda, B. Braathen, and K. Heia, Eds.), pp. 1015–1022, Norwegian Society for Image Processing and Pattern Recognition, 1993.
22. B. Kruse, An exact sequential Euclidean distance algorithm with application to skeletonizations, in *Scandinavian Conference on Image Analysis (SCIA '91)* (P. Johansen and S. Olsen, Eds.), pp. 982–992, Pattern Recognition Society of Denmark, 1991.
23. G. Sanniti di Baja, Well-shaped, stable, and reversible skeletons from the (3,4)-distance transform, *J. Visual Commun. Image Representation* **5**, 1994, 107–115.
24. G. Sanniti di Baja and Edouard Thiel, Skeletonization algorithm running on path-based distance maps, *Image Vision Comput*. **14**, 1996, 47–57.
25. C. Arcelli and G. Sanniti di Baja, Euclidean skeleton via centre-of-maximal-disc extraction, *Image Vision Comput*. **11**, 1993, 163–173.
26. C. Arcelli, Pattern thinning by contour tracing, *Comput. Graphics Image Process*. **17**, 1981, 130–144.
27. G. Sanniti di Baja and E. Thiel, (3,4)-weighted skeleton decomposition for pattern representation and description, *Pattern Recogn*. **27**, 1994, 1039–1049.
28. M. Y. Jaisimha, R. M. Haralick, and D. Dori, Quantitative performance evaluation of thinning algorithms in the presence of noise, in *Aspects of Visual Form Processing (2nd International Workshop on Visual Form)* (C. Arcelli, L. P. Cordella, and G. Sanniti di Baja, Eds.), pp. 261–286, World Scientific, Singapore, 1994.
29. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Vol. 2, Chap. 5, Addison-Wesley, Reading, MA, 1992.
30. T. Kanungo, R. M. Haralick, and I. Philips, Nonlinear global and local document degradation models, *Internat. J. Imaging Systems Technol*. **5**, 1995, 220–230.
31. G. Borgefors, I. Nyström, and G. Sanniti di Baja, Connected components in 3D neighbourhoods, in *Proceedings of 10th Scandinavian Conference on Image Analysis (SCIA '97)* (M. Frydrych, J. Parkkinen, and A. Visa, Eds.), pp. 567–572, Pattern Recognition Society of Finland, 1997.

STINA SVENSSON received the B.Sc. in mathematics from Uppsala University, Sweden, in 1996. She is currently a Ph.D. student at the Centre for Image Analysis, Swedish University of Agricultural Sciences, Uppsala, Sweden. Her research interest is method development for quantitative shape analysis of objects.

GUNILLA BORGEFORS received the M.Eng. and Lic.Eng. in applied mathematics, from Linköping University in 1975 and 1983, respectively, her Ph.D. in numerical analysis, from the Royal Institute of Technology,

Stockholm in 1986, and her Docent in image processing from Linköping University in 1992. From 1982 to 1993 she was employed at the National Defence Research Establishment, Linköping, Sweden, after some years as Director of Research for Computer Vision, and as Head of the Division of Information Systems from 1990–1993. Since 1993, she has been a full professor at Centre for Image Analysis, Swedish University of Agricultural Sciences, Uppsala, Sweden. Borgefors was President of the Swedish Society for Automated Image Analysis, 1988–1992, and Secretary and 1st Vice President of the International Association for Pattern Recognition, 1990–1994 and 1994–1996, respectively. In 1998 she become a fellow of the International Association for Pattern Recognition and a senior member of IEEE. Borgefors has published a large number of papers in international journals and conferences and has been the editor of three books on image analysis. Her current research interests are digital geometry in two, three, and higher dimensions and the application of image analysis in remote sensing and in industry.

INGELA NYSTRÖM received the M.Sc. in applied computer science and mathematics and the Ph.D. in computerized image analysis from Uppsala University, Sweden, in 1991, and in 1997, respectively. She is currently a researcher and lecturer at the Centre for Image Analysis, Uppsala, Sweden. Her area of research is method development for quantitative shape analysis of (volume) objects with applications.