Euclidean Skeletons of Digital Image and Volume Data in Linear Time by the Integer Medial Axis Transform

Hesselink, Wim H.; Roerdink, Jos B.T.M.

Published in:
leee transactions on pattern analysis and machine intelligence

DOI:
10.1109/TPAMI.2008.21

Publication date:
2008

Euclidean Skeletons of Digital Image and Volume Data in Linear Time by the Integer Medial Axis Transform

Wim H. Hesselink, *Affiliate Member*, *IEEE*, and Jos B.T.M. Roerdink, *Senior Member*, *IEEE*

Abstract—A general algorithm for computing euclidean skeletons of 2D and 3D data sets in linear time is presented. These skeletons are defined in terms of a new concept, called the *integer medial axis* (IMA) transform. We prove a number of fundamental properties of the IMA skeleton and compare these with properties of the centers of maximal disks (CMD) skeleton. Several pruning methods for IMA skeletons are introduced (constant, linear, and square-root pruning) and their properties studied. The algorithm for computing the IMA skeleton is based upon the feature transform using a modification of a linear-time algorithm for euclidean distance transforms. The skeletonization algorithm has a time complexity that is linear in the number of input points and can be easily parallelized. We present experimental results for several data sets, looking at skeleton quality, memory usage, and computation time, both for 2D images and 3D volumes.

Index Terms—Feature transform, integer medial axis, euclidean skeletonization, integer perpendicular bisector.

1 Introduction

In computer vision, skeleton generation is often one of the first steps in image description and analysis. Intuitively, a skeleton consists of the center lines of an object and, therefore, skeletons provide important structural information about image objects by a relatively small number of pixels [1].

There are four main approaches to skeletonization [2]:

1. thinning, that is, iterative removal of points from the boundary,
2. wave propagation from the boundary,
3. detection of crest points in the distance transformed image, and
4. analytical methods such as based on Voronoi diagrams.

A large number of skeletonization algorithms exist, see, i.e., [3], many of them based upon mathematical morphology [4], [5], [6], [7], [8], [9].

We note that, in algorithms of type 3, one often restricts oneself to a local maxima of the distance transform [10], but, in this way, one obtains too few skeleton points. The approach we present here is a variant of the third approach using a definition of skeletons based on Blum's medial axis transform [11].

Often, one is satisfied with approximations to the euclidean metric (for example, using chamfer metrics). In 1980, Danielsson [12] gave two good approximating euclidean distance transform (EDT) algorithms and applied them to obtain the centers of maximal (integer) disks (CMD), see below. He notes (p. 243) that the application of skeletons has been hampered by the lack of true euclidean distance maps. Especially in the 3D case, where data size can be very large, many existing algorithms for computing 3D euclidean skeletons are computationally too expensive [13]. Ge and Fitzpatrick [14] clearly identified the problem to determine the centers of maximal disks (CMD): "The problems with existing methods lie in the discrepancies between continuous and discrete image maps." The paper [14] also mentions the goal of linking the CMD into *connected* skeletons.

The main contribution of the present work is that we present a simple and easily parallelizable linear time algorithm, which computes a skeleton defined in terms of a new concept, called the *integer medial axis* (IMA) transform. The algorithm works in arbitrary dimensions and is based upon the general linear time EDT algorithm of Hirata [15], which has been rediscovered several times, that is, by ourselves, see Meijster et al. [16], and later by Maurer et al. [17], [18]. Note that our algorithm works directly on grid data (pixels in 2D, voxels in 3D). The skeletonization of polygonal data or cellular complexes is outside the scope of this paper.

The skeletonization algorithm has two phases. First, a feature transform is computed which uses essentially the same algorithm as for the distance transform, the difference being that not only are distances computed, but also background points that realize the closest distance. The actual skeletonization is performed in a second pass through the data, where the integer medial axis is computed by assigning points to the skeleton depending on their feature transform.

• *The authors are with the Institute for Mathematics and Computing Science, University of Groningen, PO Box 407, 9700 AK Groningen, The Netherlands. E-mail: {w.h.hesselink, j.b.t.m.roerdink}@rug.nl.*

Manuscript received 6 Mar. 2007; revised 5 Oct. 2007; accepted 26 Nov. 2007; published online 10 Jan. 2008.
Recommended for acceptance by R. Klette.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-03-0149.
Digital Object Identifier no. 10.1109/TPAMI.2008.21.

This paper is an extended version of our preliminary work presented at the International Symposium on Mathematical Morphology (ISMM '05) [19] and contains a thorough theoretical treatment of the mathematical properties of IMA skeletons, a more detailed discussion of skeleton postprocessing, and several examples of volume visualizations of 3D skeletons.

The IMA-skeleton in 3D will in general consist of surface and line segments. To arrive at a line skeleton in 3D (that is, a 1D centerline of a 3D object), a further extension of our method would be required, which we will consider in future work.

Often, simplification or pruning of the skeleton is used as a postprocessing step to remove unwanted points, which especially arise in noisy data [20]. In our approach, skeleton pruning is handled in the algorithm itself during the second pass of the algorithm. In our initial approach [19], this was achieved via a single adjustable parameter; in the version presented in this paper, we develop a so-called square-root pruning, which no longer involves tuning parameters and effectively suppresses artifacts due to straight boundary discretization. Another possibility would be to first extract the boundaries of image objects and then compute the shortest distances along boundary points. This leads to nicer skeletons, at the expense of increased computation time. This method was introduced by Strzodka and Telea [21], who also developed an efficient implementation on graphics hardware.

Our method does not aim at a minimal skeleton useful for image compression with exact reconstruction, but at an efficient computation of skeletons directly from the euclidean feature transform, thus avoiding the costly and complicated phase of removing centers of not-quite-maximal disks by the techniques in [22]. We establish a number of mathematical properties of the IMA and point out some relations to Blum's real medial axis (RMA) and to the CMD skeleton. The IMA-skeleton does not necessarily preserve the topology of the input image nor does the CMD skeleton (see [23] for a general discussion of digital topology). More work is needed to establish the topological characteristics of the IMA skeleton. Below, we will state a conjecture about the connectivity (in a special sense) of the unpruned IMA-skeleton. In practice, connectivity will be lost during pruning. If desired, postprocessing techniques such as connected or topology-preserving thinning can be applied to the (pruned) IMA-skeleton. The main advantage of the IMA-skeleton is that it can be computed in linear time (including the pruning step), which makes it feasible for the interactive skeletonization of large 3D data sets. It can be shown that the IMA allows perfect reconstruction of the input image, although it is not a minimal skeleton. Reconstruction issues will be discussed elsewhere.

The remainder of this paper is organized as follows: In Section 2, we briefly review our linear time feature transform computation, which forms the basis of our skeletonization method. In Section 3, we then introduce the IMA skeleton and prove a number of fundamental properties. Pruning of IMA skeletons is considered in Section 4. In Section 5, an efficient algorithm to compute the IMA skeleton is developed. For all program parts, explicit and compact pseudocode is given. Some experimental results are presented in Section 6. Section 7 contains conclusions and ideas for future work.

2 Feature Transform Computation

We briefly describe extension of the EDT algorithm to the computation of feature transforms, closely adhering to the notation and approach given in [16]. The algorithm can deal with several types of distances (Manhattan, chessboard, or chamfer distances), but we will limit ourselves to the case of the euclidean distance here since we focus on euclidean skeletons in this paper.

The length of a vector $\mathbf{r} \in \mathbb{R}^d$ is denoted by $\|\mathbf{r}\| = \sqrt{\sum_i \mathbf{r}_i^2}$. We regard $\mathbb{Z}^d$ as a grid embedded in $\mathbb{R}^d$. The elements of $\mathbb{Z}^d$ are called grid points.

A *binary image* is a pair (A, F), where A is a rectangular box of grid points and F is a subset of A, called the *foreground*. The complement $B = A \setminus F$ of F within A is called the *background* and A is called the *box*. We are mainly interested in 2D or 3D images, but the theory is most easily formulated for d-dimensional images with $d \geq 2$. The set A is thus a rectangular subset of the grid $\mathbb{Z}^d$, which itself is a subset of the euclidean vector space $\mathbb{R}^d$ with the standard inner product and associated euclidean distance.

The euclidean distance transform dt of B is the function that assigns to every grid point $\mathbf{r}$ the distance to the nearest background point, so $dt(\mathbf{r}, B) = \min\{\|\mathbf{r} - \mathbf{y}\| \,|\, \mathbf{y} \in B\}$. The *feature transform* FT is defined as the set-valued function that assigns to $\mathbf{r}$ the set of closest background points. Therefore, we have $FT(\mathbf{r}, B) = \{\mathbf{y} \in B \,|\, \|\mathbf{r} - \mathbf{y}\| = dt(\mathbf{r}, B)\}$. The parameter B is omitted from dt and FT when it is clear from the context.

It is possible to compute FT (for a linear-time algorithm, see [24]), but it is computationally cheaper and sufficient for our purposes to compute, for every point $\mathbf{r}$, just a single feature transform point $ft(\mathbf{r})$. Therefore, the function ft is incompletely specified by $ft(\mathbf{r}) \in FT(\mathbf{r})$. In fact, we compute $ft(\mathbf{r})$ as the first element of $FT(\mathbf{r})$ with respect to a lexical ordering.

The computation of ft proceeds in d phases. We specify the results of these phases as follows: For $0 < i \leq d$, let L_i be the i-dimensional subspace spanned by the first i standard basis vectors of $\mathbb{R}^d$. The ith phase computes the i-dimensional feature transform ft_i that is characterized by $ft_i(\mathbf{r}) \in FT(\mathbf{r}, B \cap (\mathbf{r} + L_i))$. Here, $\mathbf{r} + L_i = \{\mathbf{r} + \mathbf{x} | \mathbf{x} \in L_i\}$ denotes the subspace L_i translated over the vector $\mathbf{r}$. The result of the last phase is $ft = ft_d$. Since the components of $ft_i(\mathbf{r})$ orthogonal to L_i are always equal to the corresponding components of $\mathbf{r}$, we only compute and use the orthogonal projection of ft_i on L_i.

In Algorithms 1-3, we present the computation for the case $d = 3$ in a box of size (m, n, p). Since ft_i is a vector-valued function, the three components of $ft_i(\mathbf{r})$ are written $ft_i[\mathbf{r}].x$, $ft_i[\mathbf{r}].y$, and $ft_i[\mathbf{r}].z$.

Algorithm 1: Program fragment for the first phase—1D feature transform in 3D.
Input: binary image b of size $m \times n \times p$
Output: feature transform ft_1 along the first dimension
forall $y \in [0 \dots n-1]$, $z \in [0 \dots p-1]$ **do**
  $(*$ scan 1 $*)$
  if $b[m-1, y, z]$ **then** $g[m-1] := 0$
  else $g[m-1] := \infty$
  endif
  for $x := m-2$ **downto** 0 **do**
    if $b[x, y, z]$ **then** $g[x] := 0$
    else $g[x] := 1 + g[x+1]$
    endif
  end for
  $(*$ scan 2 $*)$
  $ft_1[0, y, z].x := g[0]$
  for $x := 1$ **to** $m-1$ **do**
    if $x - ft_1[x-1, y, z].x \leq g[x]$ **then**
      $ft_1[x, y, z].x := ft_1[x-1, y, z].x$
    else
      $ft_1[x, y, z].x := x + g[x]$
    endif
end forall

Algorithm 2: Program fragment for the second phase.
Input: feature transform ft_1 along the first dimension
Output: feature transform ft_2 along the second dimension
forall $x \in [0 \dots m-1]$, $z \in [0 \dots p-1]$ **do**
  $q := 0$; $s[0] := 0$; $t[0] := 0$
  for $u := 1$ **to** $n-1$ **do** $(*$ scan 1 $*)$
    while $q \geq 0 \wedge f(t[q], s[q]) > f(t[q], u)$ **do**
      $q := q - 1$
    if $q < 0$ **then**
      $q := 0$; $s[0] := u$
    else
      $w := 1 + Sep(s[q], u)$
      if $w < n$ **then**
        $q := q + 1$; $s[q] := u$; $t[q] := w$
      endif
    endif
  end for
  for $u := n-1$ **downto** 0 **do** $(*$ scan 2 $*)$
    $ft_2[x, u, z].x := ft_1[x, s[q], z].x$
    $ft_2[x, u, z].y := s[q]$
    if $u = t[q]$ **then** $q := q - 1$ **endif**
  end for
end forall

Algorithm 3: Program fragment for the third phase.
Input: feature transform ft_2 along the second dimension
Output: feature transform ft_3 along the third dimension
forall $x \in [0 \dots m-1]$, $y \in [0 \dots n-1]$ **do**
  $q := 0$; $s[0] := 0$; $t[0] := 0$
  for $u := 1$ **to** $p-1$ **do** $(*$ scan 1 $*)$
    while $q \geq 0 \wedge f(t[q], s[q]) > f(t[q], u)$ **do**
      $q := q - 1$
    if $q < 0$ **then**
      $q := 0$; $s[0] := u$
    else
      $w := 1 + Sep(s[q], u)$
      if $w < p$ **then**
        $q := q + 1$; $s[q] := u$; $t[q] := w$
      endif
    endif
  end for
  for $u := p-1$ **downto** 0 **do** $(*$ scan 2 $*)$
    $ft_3[x, y, u].x := ft_2[x, y, s[q]].x$
    $ft_3[x, y, u].y := ft_2[x, y, s[q]].y$
    $ft_3[x, y, u].z := s[q]$
    if $u = t[q]$ **then** $q := q - 1$ **endif**
  end for
end forall

The first phase is the computation of ft_1 given in Algorithm 1. For every pair (y, z), it consists of two scans over the line $(0, y, z) + L_1$. The background B is represented here by a 3D Boolean array b, i.e., $b(x, y, z) \equiv true \iff (x, y, z) \in B$. In the first scan, $g[x]$ becomes the distance to the next background point along the line. The second scan collects ft_1.

The second and third phases are given in Algorithms 2 and 3. In the body of the outer loop, the value of ft_i is computed from ft_{i-1} for a given scan line, again by two scans. The results of the forward scan are collected on stacks s and t, with common stack pointer q. The backward scan reaps ft_i as harvest. The auxiliary functions f and Sep are given by $f(i, u) = (i-u)^2 + g(u)$ and $Sep(i, u) = (u^2 - i^2 + g(u) - g(i))$ **div**$(2(u-i))$, where the function g is the squared EDT of the previous phase. Therefore, $g(i) = (x - ft_1[x, i, z].x)^2$ in phase 2 and $g(i) = (x - ft_2[x, y, i].x)^2 + (y - ft_2[x, y, i].y)^2$ in phase 3. Note that, in the body of the outer loop, we regard x and z as constants for phase 2 and x and y as constants for phase 3.

Since the algorithm is completely analogous to our algorithm for the EDT, we refer to [16] for further details.

3 Skeletonization

The feature transform of a data set can be used to compute its skeleton. We first examine the definition of the medial axis [11], see also [25], [12], [14], [22]. Actually, we present three possible formalizations: CMD, RMA, and IMA. Since RMA is not restricted to grid points, whereas CMD and IMA are, the latter two are the main contenders.

3.1 The Real Medial Axis and CMD Skeleton

For the moment, we assume that the background B is a closed subset of $\mathbb{R}^d$. For every point $x \in \mathbb{R}^d$, we can form the largest open disk $D(x, r) = \{y \in \mathbb{R}^d \mid \|x - y\| < r\}$ that is disjoint with B. This is called the *inscribed disk* of x. If an inscribed disk at point p is not contained in any other inscribed disk of B, we call it a *maximal disk* with center p. We define the RMA to consist of the points $x \in \mathbb{R}^d \setminus B$, which are centers of maximal disks.

For $x \in \mathbb{Z}^d$, the *inscribed integer disk* $M(x)$ is the intersection $D(x, r) \cap \mathbb{Z}^d$, where $D(x, r)$ is its inscribed disk. The set CMD consists of the points $x \in \mathbb{Z}^d$ for which $M(x)$ is not contained in any $M(y)$ with $y \neq x$, see

also [14], [22]. As is presumably well known, it is not true that $CMD \subseteq RMA \cap \mathbb{Z}^d$.

Example 1. Let B consist of the four points (0, 0), (3, 0), (0, 3), and (3, 3). The intersection $RMA \cap \mathbb{Z}^d$ is empty, but CMD contains the points (1, 1), (1, 2), (2, 1), and (2, 2).

Our aim is to define a skeleton that looks like the RMA of a smoothing of the background and tends to be connected when the complement of the background is connected while still being computable in linear time. In contrast, CMD is better for the compression of binary images in the sense that CMD contains fewer points, while the image still is reconstructible from CMD. We will consider image reconstruction in another publication.

Recall that $dt(x) = \min\{\|x - y\| \mid y \in B\}$ and $FT(x) = \{y \in B \mid \|x - y\| = dt(x)\}$. Clearly, $dt(x)$ is the radius of the inscribed disk of x (for $x \in B$, we regard the empty set as an open disk with radius 0). The function $ft : \mathbb{R}^d \to B$ is incompletely specified by $ft(x) \in FT(x)$.

The next lemma may not be surprising, but it seems to be new.

Lemma 2. *Assume B is a discrete (that is, locally finite) subset of $\mathbb{R}^d$. Let $x \in \mathbb{R}^d$. Then, $x \in RMA$ if and only if $FT(x)$ has more than one element.*

Proof. Suppose $FT(x)$ contains two different elements y and z. Put $r = dt(x)$. We have to prove that $D(x, r)$ is not contained in some other inscribed disk. Assume that $D(x, r)$ is contained in the inscribed disk $D(w, s)$. Then, y is a background point in the closures of both disks; therefore, both disks have the same tangential hyperplanes in y; therefore, the point x lies on the line segment from w to y. By the same argument, x also lies on the line segment from w to z. It follows that $x = w$ and $r = s$.

Conversely, assume $x \in RMA$. Then, $x \notin B$ and its inscribed disk $D(x, r)$ is not contained in any other inscribed disk. We have to prove that $FT(x)$ has more than one element. Assume that $FT(x) = \{y\}$. Since $x \notin B$, we have $x \neq y$. Since the background B is discrete, there is $s > r$ such that $D(x, s) \cap B = \{y\}$. We now choose a point w such that x lies on the line segment from w to y and that $\|w - x\| < \frac{1}{2}(s - r)$. Then, $D(w, \|w - y\|)$ is an inscribed disk that contains $D(x, r)$. This is a contradiction. □

The main point of this lemma is that it is not true when B is not discrete. For example, let B be the ellipse in $\mathbb{R}^2$ given by the equation $x^2 + a^2y^2 = a^2$ for $a > 1$. The RMA is the segment of the long axis that consists of the points $(t, 0)$ with $|t| \leq a - a^{-1}$, strictly inside the ellipse. The two extremal points $x_\pm = \pm(a - a^{-1})$ belong to RMA and yet have only one element in $FT(x_\pm)$.

Henceforth, we assume that the background consists of grid points only, that is, that $B \subseteq \mathbb{Z}^d$. It follows that B is discrete so that Lemma 2 applies. The following result is almost trivial to verify, but it is quite useful.

Lemma 3. *Let $x \in \mathbb{R}^d$ and let y, z be two different elements of $FT(x)$. Then, $\|y - z\| \geq 1$. If, moreover, $x \in \mathbb{Z}^d$, then $\|y - z\| > 1$.*

3.2 The Integer Medial Axis

Since we assume the background now to consist of grid points only, RMA contains many points that would disappear when the background is smoothed to the curved (hyper)surface in $\mathbb{R}^d$ it is supposed to represent. For example, in the case of a background that consists of the grid points of a horizontal line in $\mathbb{R}^2$, the RMA consists of the vertical lines with odd-half-integer x-coordinates. Therefore, we introduce a skeleton definition that avoids these unwanted points.

Let $E = \{e \in \mathbb{Z}^d \mid \|e\| = 1\}$. The elements of E are called *unit vectors*.

Definition 4. *The IMA consists of the points $p \in \mathbb{Z}^d$ such that, for some $e \in E$, we have $\|ft(p + e) - ft(p)\| > 1$ and $\|m - ft(p + e)\| \leq \|m - ft(p)\|$, where $m = p + \frac{1}{2}e$ is the midpoint of the line segment from p to $p + e$.*

The reason to use ft rather than FT is that ft is computationally cheaper since it involves a single closest point instead of the set of all closest points in the case of FT. Also, the restriction of FT to $\mathbb{Z}^d$ may well be single valued everywhere so that consideration of neighboring points is needed in any case. For a linear-time algorithm to compute the feature transform sets FT, see [24].

The reason for requiring the distance to be strictly greater than 1 is to avoid skeleton points that are merely due to the discreteness of the background. The second condition on the pair $(p, p + e)$ in the definition of IMA is introduced to get one point, rather than two, and specifically the point that is closest to the perpendicular bisector of the line segment from $ft(p)$ and $ft(p + e)$. If p and $p + e$ are equally close, both are included. We prefer IMA over CMD since it is easier to compute and seems to give more image information when the background is a discretization of a continuous background.

The following lemma says that IMA is disjoint with the background.

Lemma 5. $IMA \cap B = \emptyset$.

Proof. Assume $p \in IMA \cap B$. Then, $ft(p) = p$, and there is $e \in E$ with $\|ft(p + e) - p\| > 1$ and $\|m - p\| \geq \|m - ft(p + e)\|$, where $m = p + \frac{1}{2}e$. It follows that $\frac{1}{2} \geq \|m - ft(p + e)\|$ and, hence, that $\|p - ft(p + e)\| \leq 1$, a contradiction. □

The definition of IMA is primarily motivated by the next result, which shows that IMA has "enough" elements. A *Manhattan path* on $\mathbb{Z}^d$ is a sequence of grid points such that every pair of subsequent points has distance 1 (and is therefore along one of the coordinate axes).

Theorem 6. *Let p and q be points of the background B and let L be a shortest Manhattan path from p to q. If L is not contained in B, it contains a point of IMA.*

Proof. Let $r(i)$, $0 \leq i \leq k$, be the shortest Manhattan path from p to q that is not contained in B. Since it is a Manhattan path from p to q, we have $r(0) = p$, $r(k) = q$, and $\|r(i + 1) - r(i)\| = 1$ for all $0 \leq i < k$. Since the path is not contained in B, there is an index j with $0 < j < k$ and $r(j) \notin B$. Without loss of generality, we may assume that $r(1) \notin B$.

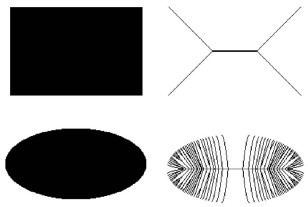

Fig. 1. Two-dimensional images with their skeletons. Left: Original images (foreground in black, background in white). Right: IMA skeleton.

Let $x(i) = ft(r(i))$ for all i. Then, $x(0) = p$ and $x(k) = q$ and $x(1) \neq r(1)$. We have $\|p - r(1)\| = 1$ and, hence, $dt(r(1)) = 1$. By Lemma 3, this implies that $x(1) = x(0)$ or $\|x(1) - x(0)\| > 1$. If x satisfies $\|x(j+1) - x(j)\| \leq 1$ for all j, then $x(1) = x(0)$, so x represents a shorter Manhattan path than r, contrary to the assumption that r is a shortest Manhanttan path. It follows that there is an index j with $0 \leq j < k$ and $\|x(j+1) - x(j)\| > 1$. Put $m = \frac{1}{2}(r(j+1) + r(j))$. If $\|m - x(j+1)\| \leq \|m - x(j)\|$, then $r(j) \in IMA$. Otherwise, $r(j+1) \in IMA$. In that case, $j + 1 < k$ because of Lemma 5. □

As an illustration of the theorem, see Fig. 1. In both cases, any interior shortest Manhattan path between discrete background points indeed intersects IMA. Note also the effect of the discreteness of the background on IMA in case of the ellipse: Instead of a central line segment, as in the continuous case, there are now a large number of line segments radiating out toward the background. The suppression (pruning) of such undesired effects will be considered in Section 4.

Although the previous theorem can be interpreted as saying that IMA has enough elements, the next result shows that IMA has not too many elements in the sense that every one of them is close to RMA.

Theorem 7. *For every $p \in IMA$, there is $e \in E$ and $t \in \mathbb{R}$ with $0 \leq t \leq \frac{1}{2}$ and $p + te \in RMA$.*

Proof. Let $p \in IMA$. Then, there is $e \in E$ with $\|ft(p) - ft(p+e)\| > 1$ and $\|m - ft(p)\| \geq \|m - ft(p+e)\|$, where $m = p + \frac{1}{2}e$. First, assume that $ft(p) \in FT(m)$. Then, $ft(p)$ is the closest point on B to m. Therefore, $\|m - ft(p)\| \leq \|m - ft(p+e)\|$. Since $\|m - ft(p)\| \geq \|m - ft(p+e)\|$, it follows that $\|m - ft(p)\| = \|m - ft(p+e)\|$ and that both $ft(p)$ and $ft(p+e)$ are elements of $FT(m)$. In view of Lemma 2, this implies that $m \in RMA$ is a point as looked for. It remains to treat the case with $ft(p) \notin FT(m)$. Let point z be the last point of the line segment from p to m with $ft(p) \in FT(z)$. By continuity, this point exists. Since $ft(p) \notin FT(z')$ for points z' arbitrary close to z, the set $FT(z)$ consists of more than one element. Therefore, $z \in RMA$. □

The converse of this theorem is not true. The simplest counterexample in $\mathbb{R}^2$ is the case mentioned at the beginning of this section: If $B = \{(x, 0) | x \in \mathbb{Z}\}$, then IMA is empty and RMA equals all perpendicular bisectors of neighboring pairs.

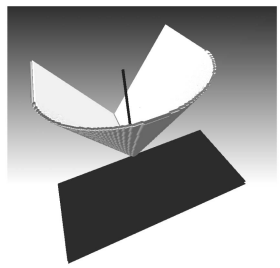

Fig. 2. Example: When the background equals the union of a halfplane and orthogonal halfline (black color), the IMA is the union of a half-cone and two quarterplanes (white/gray color). See text for details.

3.3 Connectivity of IMA versus CMD

As illustrated by Theorem 6, IMA has some good connectivity properties. In general, we conjecture that, if the complement of B is bounded and connected, then IMA is connected with respect to 8-connectivity in $\mathbb{Z}^2$ or, more generally, sq2-connectivity for $\mathbb{Z}^d$ (see Section 3.4). In that respect, it is better than CMD.

Example 8. Let B be the intersection of $\mathbb{Z}^2$ with the union of the x-axis and the y-axis. Then, IMA consists of the points $(x, \pm x)$ for all $x \in \mathbb{Z} \setminus \{0\}$ and CMD is a subset of IMA that contains $(\pm 3 \pm 3)$ and $(\pm 4, \pm 4)$, but misses at least the points $(\pm 1, \pm 1)$, $(\pm 2, \pm 2)$, and $(\pm 5, \pm 5)$.

One might conjecture that, for every point $p \in CMD \setminus IMA$, there is an $e \in E$ with $p + e \in IMA$. This, however, is not true, as is shown by the following example:

Example 9. Consider in $\mathbb{Z}^2$ the image with the background B that consists of the four points $(83, 14)$, $(-14, 83)$, $(-83, -14)$, and $(14, -83)$ (note the rotational symmetry). It turns out that $p = (4, 1)$ is an element of CMD. The points of IMA nearest to p are $(2, 3)$ and $(2, -1)$, both at distance $2\sqrt{2}$.

Although the CMD is not always contained in the IMA, we will describe elsewhere that the IMA allows perfect reconstruction of the input image.

The complement of $IMA \cup B$ can be Manhattan-connected in nontrivial ways. The simplest case is $B = \{(2, 2), (3, 2), (2, 3)\}$ in $\mathbb{Z}^2$. Here, IMA consists of the integer solutions of $x = y \geq 3$. A more complicated example in $\mathbb{Z}^3$ uses the background $B = H \cup L$, where H is the halfplane of the points (x, y, z) with $z = 0 \leq x$ and L is the halfline of the points with $x = y = 0 \leq z$. In this case, IMA is the union $C \cup Q_{\pm}$, where C is an approximation of the half-cone of the real points with $x^2 + y^2 = z^2 \wedge z \geq 1 \wedge x \geq 0$ and the two quarterplanes $Q_{\pm}$ of the solutions of $1 \leq z = \pm y \wedge x \leq 0$; see Fig. 2. Since the quarterplanes $Q_{\pm}$ do not touch, the complement of $IMA \cup B$ is Manhattan connected.

3.4 The Integer Perpendicular Bisector

In this section, we analyze IMA for the case where the background has precisely two elements.

An easy linear algebra calculation shows that, for all p, x, and $y \in \mathbb{R}^d$, we have

$$\|p - x\| \geq \|p - y\| \equiv (y - x, 2p - x - y) \geq 0. \qquad (1)$$

Here, $(\cdots,\cdots)$ is used to denote the standard inner product of $\mathbb{R}^d$. The equivalence remains valid when $\geq$ is replaced by $\leq$ or $=$. In particular, the "real" perpendicular bisector of the line segment from x to y consists of the vectors p with $(y-x, 2p-x-y)=0$. If we want a reasonable number of grid points on the bisector, we have to relax the condition that the inner product should be 0. This is done in the following definition:

Recall that the L_∞ norm on $\mathbb{R}^d$ is given by $\|x\|_\infty = \max_i |x_i|$. We define the *integer perpendicular bisector* of the line segment between grid points x and y to consist of the grid points $p \in \mathbb{Z}^d$ with $|(y-x, 2p-x-y)| \leq \|y-x\|_\infty$.

Lemma 10. *Let x and y be grid points with $\|x-y\|>1$. Let $B=\{x,y\}$. Then, IMA equals the integer perpendicular bisector of the line segment between x and y.*

Proof. Write H for the integer perpendicular bisector of the line segment between x and y. We first prove $H \subseteq IMA$. Let $p \in H$. Without loss of generality, we may assume that $ft(p)=y$. It follows that $\|p-x\| \geq \|p-y\|$. By (1), we have $(y-x, 2p-x-y) \geq 0$. Since $p \in H$, we have $0 \leq (y-x, 2p-x-y) \leq \|y-x\|_\infty$. There is an index i such that $|y_i - x_i| = \|y-x\|_\infty > 0$. This implies that there is a unit vector $e \in E$ with $(y-x, e) = -\|y-x\|_\infty < 0$. It follows that $(y-x, 2p+e-x-y) \leq 0$ and also $(y-x, 2p+2e-x-y) < 0$. By (1), this implies that $m = p + \frac{1}{2}e$ satisfies $\|m-x\| \leq \|m-y\|$ and that $\|p+e-x\| < \|p+e-y\|$. We therefore have $ft(p+e)=x$ and $p \in IMA$.

Conversely, let $p \in IMA$. Without loss of generality, we may assume that $ft(p)=y$. Since $p \in IMA$ and $B=\{x,y\}$, there is $e \in E$ with $ft(p+e)=x$ and $\|m-x\| \leq \|m-y\|$ for $m = p+\frac{1}{2}e$. Since $\|p-x\| \geq \|p-y\|$, (1) implies that $(y-x, 2p-x-y) \geq 0$. The formula about m yields $(y-x, 2p+e-x-y) \leq 0$ and, hence, $0 \leq (y-x, 2p-x-y) \leq (y-x, e) \leq \|y-x\|_\infty$. This proves that $p \in H$. □

In particular, this proves that, in this special case, the set IMA does not depend on the choice of function ft (see also Section 3.5).

We define an *sq2-edge* in $\mathbb{Z}^d$ to be a pair of grid points $p, q \in \mathbb{Z}^d$ with $\|p-q\| \leq \sqrt{2}$. Note that, in 3D, this corresponds to 18-connectedness. We define an *sq2-path* to be a sequence of grid points such that subsequent pairs are sq2-edges. A subset of $\mathbb{Z}^d$ is called *sq2-connected* if and only if every pair of elements can be connected via an sq2-path. In $\mathbb{Z}^2$, sq2-connectivity is the same as 8-connectivity.

Lemma 11. *The integer perpendicular bisector of a line segment between x and y in $\mathbb{Z}^d$ is sq2-connected.*

Proof. Without loss of generality, we may assume that $x=0$ and that $\|y\|_\infty$ equals the first coordinate y_1 of y. Let H be the integer perpendicular bisector of x and y. Then, H consists of the grid points p with $|(y, 2p-y)| \leq y_1$. Let H' be the subset of H that consists of the grid points p with $-y_1 \leq (y, 2p-y) < y_1$ (note that the left-hand relation is $\leq$, whereas the right-hand relation is $<$). If $p \in H \setminus H'$, then $(y, 2p-y) = y_1$ so that $p - e_1 \in H'$ for the first basis vector e_1 of $\mathbb{Z}^d$. Therefore, every point $p \in H \setminus H'$ has an sq2-edge to H'. It remains to show that H' is sq2-connected.

Let $f : \mathbb{Z}^d \to \mathbb{Z}^{d-1}$ be the projection function that removes the first coordinates. It is easy to see that f induces a bijection $H' \to \mathbb{Z}^{d-1}$. Let $g : \mathbb{Z}^{d-1} \to H'$ be the inverse bijection. If $\|q-q'\|=1$ in $\mathbb{Z}^{d-1}$, then $\|g(q)-g(q')\| \leq \sqrt{2}$. Since $\mathbb{Z}^{d-1}$ is Manhattan connected, it follows that H' is sq2-connected. □

We now formulate the following conjecture.

Conjecture 12. *If the complement of B is bounded and sq2-connected, then IMA is sq2-connected.*

Remark 13. In general, IMA has codimension 1, but holes in the background can generate pieces of IMA of higher codimension, as the next example shows. Consider in $\mathbb{Z}^3$ the background that consists of the 20 grid points x with $\|x\|_\infty = 1 < \|x\|$. Then, IMA is the union of the three coordinate axes and is of codimension 2.

3.5 How Much Does IMA Depend upon the Choice of ft?

A disadvantage of IMA is that it can (weakly) depend on the choice of function ft within FT. The following is an example.

Example 14. Take $d=2$. Let the background B consist of the three points $a=(-6,7)$, $b=(3,9)$, and $c=(10,2)$. We claim that, in this case, the question of whether $0 \in IMA$ holds depends on the choice of ft.

We have the unit vectors $e=(1,0)$, $f=(0,1)$, and $-e$ and $-f$. It is easy to verify that $FT(e)=\{b,c\}$ and that $FT(x)=\{a\}$ for each $x \in \{0,-e,f,-f\}$. It follows that $ft(e)=b$ or c and that $ft(x)=a$ for each $x \in \{0,-e,f,-f\}$. The neighbors $x \neq e$ of the origin cannot be used to infer $0 \in IMA$ since they have $ft(x)=ft(0)$. Therefore, since $\|b-a\| > 1$ and $\|c-a\|>1$, we get that $0 \in IMA$ if and only if $\|m-ft(e)\| \leq \|m-a\|$, where $m=\frac{1}{2}e$. It turns out that $\|m-a\|=9.55$, $\|m-b\|=9.34$, and $\|m-c\|=9.71$. This implies $0 \in IMA$ if $ft(e)=b$ and $0 \notin IMA$ if $ft(e)=c$.

We will now show that, when the choice of ft within FT is replaced by another, the set IMA does not change very much. We first prove an auxiliary result about the case that neighboring grid points have neighboring feature transforms. For a vector $e \neq 0$, we write $e^\perp$ to denote the hyperplane $\{z \in \mathbb{R}^d \mid z \perp e\}$. (Recall that "$\perp$" stands for orthogonality.)

Lemma 15. *Let p and q be grid points with $\|p-q\|=1$. Assume that $x \in FT(p)$ and $y \in FT(q)$ satisfy $\|x-y\|=1$. Write $e=q-p$. Then, $y=x+e$ and $x-p \in e^\perp$.*

Proof. We have $e \in E$. By translation, we may assume that $p=0$ and $q=e$. Write $f=y-x$. Then, $f \in E$. Since x and $y=x+f$ are both background points and x and f are grid points, we have

$$x \in FT(0) \quad \Rightarrow \quad \|x\| \leq \|x+f\| \quad \Rightarrow \quad 2(x,f)+1 \geq 0 \quad \Rightarrow \quad (x,f) \geq 0.$$

A similar calculation yields

$$x+f\in FT(e)\Rightarrow\|x-e+f\|\le\|x-e\|$$
$$\Rightarrow 2(x-e,f)+1\le 0\Rightarrow(x,f)\le(e,f)-1.$$

Since e and f are elements of E, we have $(e,f)\le 1$, with equality if and only if $e=f$. This implies that $e=f$ and $(x,e)=0$. □

Lemma 16. *Let $p\in\mathbb{Z}^d$ and $y\in FT(p)$ and $e\in E$ be such that $(y-ft(p),e)>0$. Then, we have $p\in IMA$ or $p+e\in IMA$.*

Proof. Without loss of generality, we may assume that $p=0$. Write $x=ft(0)$. Then, we have $\|y\|=\|x\|$. Since x and y are integral, we have $(y-x,e)\ge 1$. Since $\|x\|=\|y\|$, we then have $\|x-e\|^2-\|y-e\|^2=2(y-x,e)\ge 2$. Since y is a background point, this implies that $x\notin FT(e)$.

We write $z=ft(e)$. Then, $\|x-e\|^2-\|z-e\|^2\ge 2$ and, hence, $x\ne z$. If $\|z-x\|=1$, then Lemma 15 implies that $\|x-e\|^2-\|z-e\|^2=1$ by the Theorem of Pythagoras. This proves that $\|x-z\|>1$ and, hence, $0\in IMA$ or $e\in IMA$ by the definition of IMA. □

For every nonzero vector v, there is an element $e\in E$ with $(v,e)>0$. Therefore, Lemma 16 implies:

Corollary 17. *Assume that p is a grid point with $\#FT(p)\ge 2$. Then, $p\in IMA$ or $p+e\in IMA$ for some $e\in E$.*

The next result shows that IMA does not depend strongly on the choice of ft.

Theorem 18. *Let $ft1$ and $ft2$ be two different choice functions within FT. Let $IMA1$ and $IMA2$ be the corresponding sets IMA. Assume that $p\in IMA1$. Then, $p\in IMA2$ or $p+e\in IMA2$ for some $e\in E$.*

Proof. Without loss of generality, we may assume that $p=0$. If $FT(0)$ has more than one element, Corollary 17 applied to $IMA2$ yields the assertion. We may therefore assume that $FT(0)$ is a singleton set, say, $FT(0)=\{x\}$.

Then, $ft1(0)=ft2(0)=x$. Since $0\in IMA1$, there is $e\in E$ such that $y=ft1(e)$ satisfies $\|x-y\|>1$ and $\|\frac{1}{2}e-x\|\ge\|\frac{1}{2}e-y\|$. By (1), this implies that $(y-x,e-x-y)\ge 0$ and, hence, $(y-x,e)\ge\|y\|^2-\|x\|^2$. Since $y\notin FT(0)$, we have $\|x\|<\|y\|$. This proves

$$0<\|y\|^2-\|x\|^2\le(y-x,e). \tag{2}$$

It follows that $\|y\|^2-\|x\|^2<2(y-x,e)$ or, equivalently, $\|e-y\|<\|e-x\|$. This implies that $x\notin FT(e)$. If $\|x-ft2(e)\|>1$, then 0 or e is in $IMA2$ and we are done. We may therefore assume that $\|ft2(0)-ft2(e)\|\le 1$, that is, $\|x-ft2(e)\|\le 1$.

Since $x\notin FT(e)$, it follows that $\|x-ft2(e)\|=1$. Therefore, Lemma 15 implies that $ft2(e)=x+e$ and $x\in e^\perp$. Since y and $x+e$ are both in $FT(e)$, we have $\|x\|=\|y-e\|$. Substitution in (2) yields $0<\|y\|^2-\|y-e\|^2\le(y,e)$ or, equivalently, $0<2(y,e)-1\le(y,e)$. Since (y,e) is an integer, it follows that $(y,e)=1$.

We now define $z=y-e$ so that $\|z\|=\|x\|$ and $z\in e^\perp$. Since $\|x-y\|>1$, we have $z\ne x$. Since $\|z\|=\|x\|$, it follows that $(x,z-x)\ne 0$. Consider the set of unit vectors $E'=\{f\in E\,|\,(z-x,f)\ge 1\}$. Since e, $-e\notin E'$, the set E' is an orthonormal set in $e^\perp$ and $z-x=\sum(z-x,f)f$, where f ranges over E'. Since $(x,z-x)\ne 0$, there is a unit vector $f\in E'$ with $(x,f)\ne 0$.

We shall prove that $\|ft2(0)-ft2(f)\|>1$. We first verify that

$$\begin{aligned}&\|f-x\|>\|f-y\|\\ \equiv&\{\text{formula}(1)\}\quad(y-x,2f-x-y)>0\\ \equiv&\{z=y+e\}\quad(z+e-x,2f-x-z-e)>0\\ \equiv&\{x,z,f\in e^\perp\}\quad(z-x,2f-z-x)-(e,e)>0\\ \equiv&\{\|z\|=\|x\|,\ \|e\|=1\}\quad 2(z-x,f)-1>0\\ \equiv&\{f\in E'\}\quad true.\end{aligned}$$

Since $y\in B$, this implies that $x\notin FT(f)$ and, hence, $x\ne ft2(f)$. Recall that $x=ft2(0)$. If $\|x-ft2(f)\|=1$, Lemma 15 implies $x\perp f$, contradicting $(x,f)\ne 0$. This proves that $\|ft2(0)-ft2(f)\|>1$. Therefore, 0 or f is an element of $IMA2$. □

4 Pruning of IMA

For many images, both natural and artificial, IMA contains many "unwanted" points: points that a human observer would not regard as belonging to a natural skeleton. Therefore, when the medial axis is used for image analysis, it is often useful to prune it of disturbing details in some postprocessing phase. Our construction of the IMA yields some information that is very useful for this purpose.

4.1 Constant Pruning

The easiest pruning, as proposed by us in [19], is to strengthen the condition $\|ft(p)-ft(p+e)\|>1$ in the definition of IMA by replacing ">1" by "$>\gamma$" for some pruning parameter $\gamma\ge 1$. This removes some points of IMA that are due to staircasing of the discrete background.

Example 19. Let $d=2$. Let $n\in\mathbb{N}$ be given. Consider B consisting of the grid points (x,y) with $|x|+|y|>n$. The complement of B in $\mathbb{Z}^2$ is bounded and connected. IMA consists of the two coordinate axes together with the checkerboard points (x,y) with $n-|x|-|y|$ even. The latter points are on the perpendicular bisectors of neighboring points on the background. These points disappear if one uses a pruning parameter $\gamma\ge\sqrt{2}$.

When the background is given by $|x|+2\cdot|y|>n$, one needs $\gamma\ge\sqrt{5}$ to remove the artifacts of the skew lines.

In this way, IMA points for which the closest background points are too close together are pruned. Let us call this method *constant pruning*. For many practical images, one can find a value for γ such that $IMA(\gamma)$ is an acceptable skeleton. The best value for γ depends on the coarseness of the image. On the one hand, a small value of γ leads to "unwanted" skeleton points due to the discretization of the background B. On the other hand, if one wants a skeleton that traverses a channel with a width of k pixels, one needs to take $\gamma<k$.

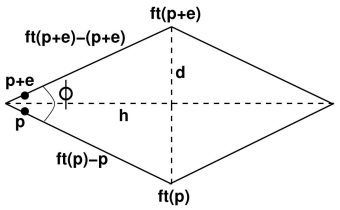

Fig. 3. Linear pruning is equivalent to setting a minimum on the bisector angle ϕ.

4.2 Linear Pruning

For many images, one would want to apply a pruning constant γ that varies over the different regions of the image, namely, small where dt is small and large where dt is large.

This suggests taking γ proportional to dt. More precisely, for any $\gamma \in \mathbb{N}$, let $IMA[\gamma]$ consist of the points $p \in A$ such that, for some $e \in E$, we have $\|ft(p) - ft(p+e)\| > \frac{1}{2}\gamma\|ft(p) - p + ft(p+e) - (p+e)\|$ and $\|m - ft(p)\| \geq \|m - ft(p+e)\|$, where $m = p + \frac{1}{2}e$. Let us call this method *linear pruning*.

Referring to Fig. 3, we observe that, roughly speaking, $ft(p)$ and $ft(p+e)$ form the equal legs of an isosceles triangle with top angle ϕ, where

$$\tan\left(\frac{1}{2}\phi\right) = d/h =$$

$$\|ft(p) - ft(p+e)\|/\|ft(p) - p + ft(p+e) - (p+e)\| > \frac{1}{2}\gamma.$$

For small ϕ, this reduces to the condition $\phi > \gamma$. Therefore, γ is a measure for the critical angle at which linear pruning gets effective. Since we always have the additional condition that $dif := \|ft(p) - ft(p+e)\|^2 > 1$ and dif is integer, we see that linear pruning is only effective when $\frac{1}{4}\gamma^2\|ft(p) - p + ft(p+e) - (p+e)\|$ is at least equal to 2, i.e., at a certain distance from the background. For example, if $\gamma = \frac{1}{3}$, linear pruning only becomes effective when $\|ft(p) - p + ft(p+e) - (p+e)\|^2 \geq 72$, i.e., at a distance of at least $\frac{1}{2}\sqrt{72} = 4.5$ pixels from the background.

Linear pruning is more or less equivalent to removing all points from the medial axis that have a bisector angle smaller than some constant, as is used in Couprie and Zrour [26]. It is not precisely the same since, for reasons of efficiency, we consider all unordered pairs of neighboring pixels, whereas Couprie and Zrour consider all pixels with all their neighbors (this gives a factor of 2), and we consider specific feature points, whereas they use all elements of the feature set (called "downstream").

4.3 Square-Root Pruning

Pruning the skeleton of points with a bisector angle less than a fixed angle, say, ϕ_0 (or, equivalently, taking γ proportional to dt), has the following disadvantage. Assume (in 2D) that the background is the union of two halfplanes with an angle $\pi - \phi$, where $\phi < \phi_0$. Then, every point of the bisectrix of the angle has two feature transform points on the boundary lines of the halfplanes with a bisector angle ϕ. Since $\phi < \phi_0$, the whole bisectrix is eliminated from the skeleton. This is unsatisfactory since this bisectrix is the RMA as defined in Section 3.1.

The remedy we propose is to let γ grow slower than dt, roughly speaking proportional to the square root of dt. In the case of an obtuse angle between two halfplanes, this would have the effect that only the IMA points close to the top are pruned from the bisectrix. The choice to take a factor like the square root of dt is justified as follows:

A *halfspace* of $\mathbb{R}^d$ is a subset of the form $H = \{X \in \mathbb{R}^d \mid (u, x) \geq c\}$ for some $u \in \mathbb{R}^d$ and some $c \in \mathbb{R}$. An integral halfspace is an intersection $H \cap \mathbb{Z}^d$, where H is a halfspace. We now want to prune our definition of IMA in such a way that the IMA is empty whenever the background is an integral halfspace and not more than necessary for this purpose.

In order to get a good pruning value, we need to know, for any pair of neighboring grid points, the maximal distance between feature transform points in any halfspace, expressed in something like their distance transforms. We do not completely know the answer, but we have a reasonably supported guess and a fully supported partial answer. We first present the guess.

Conjecture 20. *There is a constant C_d, depending only on the dimension d, such that, for every integral halfspace B of $\mathbb{R}^d$, every $p, q \in \mathbb{Z}^d$, $x \in FT(p)$, and $y \in FT(q)$ with $\|p - q\| = 1$, we have*

$$\|x - y\|^2 \leq 2 \cdot (p - q, x - y) + \|x - p + y - q\| + C_d. \quad (3)$$

The critical issue is the occurrence of a square to the left of the inequality sign and the absence of squares to the right.

The conjecture was proven for $d = 2$ with minimal value $C_2 = 2 - \sqrt{2}$ by Van der Kallen [27], who also has indications that the inequality holds for $d = 3$ with $C_3 = 1.5$.

In view of Conjecture 20, we define $IMA\text{-}SRP$ to consist of the points $p \in A$ such that, for some $q \in A$, we have $\|p - q\| = 1$ and $\|ft(p) - ft(q)\| > 1$ and

$$\|ft(p) - ft(q)\|^2 > 2 \cdot (p - q, ft(p) - ft(q)) + \|ft(p) + ft(q) - p - q\| + C_d \quad (4)$$

and $\|m - ft(p)\| \geq \|m - ft(q)\|$, where $m = \frac{1}{2}(p + q)$. The definition immediately implies that $IMA\text{-}SRP \subseteq IMA$. The resulting skeleton is fairly insensitive to the choice of C_d. For $d = 2$, we take $C_2 = 1$, being the integer closest to the minimal value $C_2 = 2 - \sqrt{2}$.

The application of formula (4) in the definition of $IMA\text{-}SRP$ is called *square-root pruning*. The pruning precludes $IMA\text{-}SRP$ from traversing a very narrow channel, like a single Manhattan path. $IMA\text{-}SRP$ does traverse a channel, however, if it is wide enough for two pawns to walk abreast. $IMA\text{-}SRP$ is not necessarily connected. For example, in 2D, if the image consists of the points (0, 0), (4, 0), (0, 20), and (4, 20), $IMA\text{-}SRP$ consists of three segments: the short segments of the points $(2, t)$ with $|t| \leq 4$ and $|t - 20| \leq 4$ and the long segment of the points $(t, 10)$ with $|t - 2| \leq 181$.

5 Implementation

5.1 Computation of IMA

The program code for the skeletonization step for $d = 3$ is shown in Algorithm 4. We work with squared distances instead of distances, which avoids the computation of square roots and thus saves time.

Algorithm 4: Program fragment for computing the IMA skeleton from the feature transform. The *pruning condition* in the **compare** procedure is given in (5)-(7).

```
procedure IMA skeleton
Input: feature transform ft_3[i,j,k] of input data input[i,j,k]
of size m × n × p
Output: skeleton skel[i,j,k] of size m × n × p
define WHITE 1 (* foreground value *)
define BLACK 0 (* background value *)
define SKEL −1 (* skeleton value *)
for i := 0 to m − 1 do
   for j := 0 to n − 1 do
      for k := 0 to p − 1 do
         if (i > 0 and (input[i,j,k] = WHITE or
                        input[i − 1,j,k] = WHITE))
            then compare(i, j, k, i - 1, j, k) endif
         if (j > 0 and (input[i,j,k] = WHITE or
                        input[i,j − 1,k] = WHITE))
            then compare(i, j, k, i, j - 1, k) endif
         if (k > 0 and (input[i,j,k] = WHITE or
                        input[i,j,k − 1] = WHITE))
            then compare(i, j, k, i, j, k - 1) endif
      end for
   end for
end for

procedure compare(i, j, k,p,q, r)
x := [i,j,k]; y := [p,q,r]
x_f := ft_3[x]; y_f := ft_3[y]
if ||x_f − y_f||^2 > 1 and pruning condition then
   crit := inprod(x_f − y_f, x_f + y_f − x − y)
   if crit ≥ 0 then skel[x] := SKEL
   endif
   if crit ≤ 0 then skel[y] := SKEL
   endif
endif
```

The constant SKEL in the procedure **IMA skeleton** of Algorithm 4 can be chosen arbitrarily as long as it is distinct from WHITE and BLACK. Also, we have added a test in this procedure that checks whether at least one of the points (i, j, k) and (p, q, r) that are tested in the procedure **compare** is a foreground point in the input. We will refer to this as "optimized skeletonization." The rationale for this is that a point that belongs to the background can never be a skeleton point. If one omits this test ("nonoptimized skeletonization"), the skeleton one obtains is the same, but the computation time for the skeletonization step will be (much) larger if the input contains many background points.

Another optimization that saves memory is to use an *in-place* calculation of the skeleton points, that is, to modify the value of a point (i, j, k) in the input array from WHITE to SKEL when the algorithm has determined that (i, j, k) is a skeleton point. If desired, one may always, in a postprocessing step, put all remaining (that is, nonskeleton) foreground points in the input image to background.

The *pruning condition* in the **compare** procedure is given by the following equations, depending on the type of pruning chosen:

- *Constant pruning:*

$$\|x_f - y_f\|^2 > \gamma^2. \tag{5}$$

- *Linear pruning:*

$$\|x_f - y_f\|^2 > \frac{1}{4}\gamma^2 \|x_f + y_f - x - y\|^2. \tag{6}$$

- *Square-root pruning:*

$$\|x_f - y_f\|^2 \geq 2 \cdot \text{inprod}(x - y, x_f - y_f) + \|x_f + y_f - x - y\| + C_d. \tag{7}$$

We will refer to the IMA-skeleton with constant, linear, and square-root pruning as IMA-CP, IMA-LP, and IMA-SRP, respectively.

For 2D images, the code has to be trivially adapted by omitting the loop over the third spatial index in the **skeleton** code fragment and using $x := [i, j]$; $y := [p, q]$ in the **compare** code fragment. In the case of square-root pruning for $d = 2$, we take $C_2 = 1$, for $d = 3$, we take $C_3 = 1.5$ (see Section 4.3).

6 Experimental Results

We have run the skeletonization algorithm on several data sets, looking at skeleton quality, memory usage, and computation time, both for 2D images and 3D volumes.

All performance measurements were carried out on a PC with a 2.8 GHz Pentium processor with a level-2 cache size of 1,024 Kbytes and 1,024 Mbytes of internal memory.

6.1 Skeletonization Quality

6.1.1 2D Skeletons

To get an idea of the quality of our skeletonization algorithm, we first give a number of examples of 2D skeletons, see Fig. 4. For comparison, we also show the CMD skeletons.

As can be seen from the results, the nonpruned IMA output is very sensitive to discretization, leading to extended barbs starting at the background. Constant pruning of IMA with γ sufficiently high effectively suppresses these barbs. The value of γ to be used will in general depend on the input image. In the examples, we have chosen γ such that the major skeleton branches (branches that approximate the RMA after smoothing of the boundary) remain intact and connected. This means that, for images with small canals, such as the angiographic image in Fig. 4 (lowest row), the value of γ will be smaller than for an image with many straight boundaries, like in the first row in Fig. 4. For linear pruning, we observe the effect

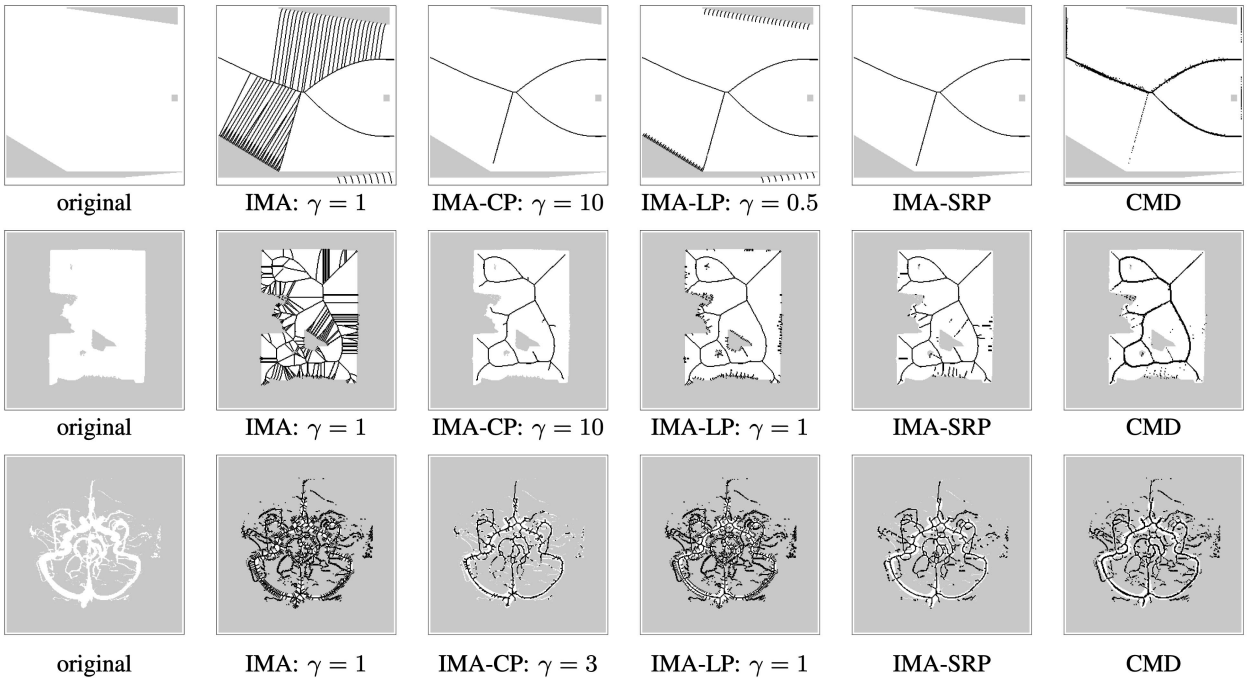

Fig. 4. Two-dimensional images with their skeletons. From left to right: Original image (foreground in white, background in gray); IMA skeleton; IMA skeleton, constant pruning; IMA skeleton, linear pruning; IMA skeleton, square-root pruning; and CMD skeleton.

already discussed in Section 4: Close to the background, no pruning takes place. This can be improved somewhat by increasing γ, but again at the cost of disturbing the main skeleton branches. Square-root pruning has the advantage over the other two pruning methods that no parameter has to be set, without any substantial additional computational costs. It generally has much fewer background artifacts than linear pruning. For data with straight boundaries, it is as good as constant pruning with a suitably chosen parameter value. For images with more curved details, constant pruning with a suitably chosen value of γ can give a result that is visually more pleasing than that of square-root pruning in the sense that the skeleton has fewer small branches. The CMD skeletons also have some background artifacts, as well as a small amount of noise along the main skeleton branches. With some pruning, this could be removed. Pruned IMA skeletons, as well as CMD skeletons, may be disconnected. If desired, connected skeletons can be obtained by postprocessing techniques such as connected thinning. If additional topological characteristics are required, then topology-preserving thinning should be employed.

6.1.2 3D Skeletons

For the 3D case, the IMA skeleton will consist of surface patches. To get insight into the structure of these skeleton surfaces, we will use volume-rendering techniques with transparency and color [28].

The experiments were performed for several 3D data sets: 1) CT scan of a tooth (source: The Volume Library[1]), 2) an angiographic data set and a scan of an engine,[2] and 3) a "horse" data set (a polygonal surface model which was scan converted to a voxel representation at several resolutions; source: Large Geometric Model Archive, Georgia Institute of Technology[3]).

The volume data sets were first converted to binary format by thresholding them with a small threshold value. This operation also removed some background noise. The sizes of the data sets are given in Table 1.

As we have seen for the 2D case, linear pruning leads to skeletons that have many small barbs at the object boundaries. Although this may still be acceptable in 2D, this no longer holds for 3D skeletons. In this case, linear pruning will give rise to many small extrusions at the skeleton surface of the input volume. These "surface barbs" form a dense surface that obstructs the view of the major skeleton surface inside the object volume. Therefore, in the next examples, we only consider constant pruning and square-root pruning.

For the "horse" and "tooth" data sets, the skeletons are visualized in Figs. 5 and 6, respectively. We show three views from different viewing angles. For reference, the input surface is rendered partially transparent in the skeleton renderings. Constant pruning with $\gamma = 7$ gives quite good results, with fewer small surface barbs than in the case of square-root pruning. For larger values of γ, the skeleton becomes heavily disconnected.

For the tooth data set, we also show some slices through the 3D skeletons in Fig. 7 at several locations in the volume. Note that these images represent slices through a 3D skeleton; therefore, they only give an indication of skeleton quality.

We conclude that our IMA algorithm with appropriately chosen pruning strategies can give surface skeletons that display major shape characteristics of binary volume data.

1. http://www9.informatik.uni-erlangen.de/External/vollib.
2. http://www.volvis.org.
3. http://www-static.cc.gatech.edu/projects/large_models.

TABLE 1
Timing Results for Feature Transform, Nonoptimized IMA Skeletonization (Skeleton), and Optimized IMA Skeletonization (Skeleton-Opt) Steps for Several 3D Data Sets

Data	Dimensions	Size	PFP	Feature transform	Skeleton	Skeleton-Opt
engine	256×256×128	8.388.608	16.1	1.99	2.50	0.51
vessels	256×256×256	16.777.216	0.67	4.56	4.86	0.24
tooth	256×256×161	10.551.296	45.6	2.17	3.22	0.13
horse-1	128×128×128	2.097.152	10.9	0.37	0.62	0.09
horse-2	128×128×256	4.194.304	11.0	0.95	1.24	0.19
horse-3	256×256×256	16.777.216	11.0	4.62	4.98	0.75
horse-4	256×256×512	33.554.432	11.1	10.14	9.95	1.49
horse-5	512×512×256	67.108.864	11.1	24.66	19.81	3.02

CPU times (seconds) are averages over 10 runs. Data dimensions, total input size (bytes), and percentage of foreground points (PFP) are indicated.

6.2 Memory Usage

The algorithm consists of two steps, that is, the feature transform and the skeletonization step. As is apparent from the code fragments in Algorithms 1-3, the feature transform algorithm is not very cache friendly. This means that the algorithm will slow down as soon as the required data do not fit in the cache. Another transition occurs when the data no longer fit in internal memory. Therefore, in our timings experiments, we first consider the case of 2D images of various sizes that all fit in cache memory. Thereafter, we look at a number of large 3D data sets for which this is not the case.

The required memory for the complete algorithm can be determined as follows: cf., Algorithm 4. Recall that m, n, and p are the dimensions of the input volume. First, we need to load the data. Since the input is a binary volume, we can use one byte for each volume element, requiring $m \cdot n \cdot p$ bytes of storage. In the feature transform routine, three arrays, $ft_1[x, y, z]$, $ft_2[x, y, z]$, and $ft_3[x, y, z]$, of type integer are used (we need integers to encode the spatial position of the feature transform points). One can check that we need at most two of these at any given time during execution (the array ft_3 can be used to hold the final result). We also need room for three 1D arrays, $g[\,]$, $s[\,]$, and $t[\,]$, of type integer, with dimension $N = \max(m, n, p)$. The values in the feature transform can be stored as integers. Therefore, we need storage for $2m \cdot n \cdot p + 3\,N$ integers. For the output array $skel[x, y, z]$, one needs, in principle, another $m \cdot n \cdot p$ bytes, but we always use in-place calculation (see Section 5.1), where the input array is used for holding the skeletons points. Adding this up, we find that we need storage for a total of $m \cdot n \cdot p$ bytes and $2m \cdot n \cdot p + 3\,N$ integers.

When the input is 2D ($m = n = N$, $p = 1$), we found that the required storage does not exceed the cache size on our system as long as N is not much larger than 200. For the 3D case, the size of the data we used always exceeded the cache size. However, as long as the product $m \cdot n \cdot p$ is not larger than around 50 million points, the total storage requirement does not exceed the internal memory of 1,024 Mbytes.

6.3 Computation Time

6.3.1 2D Skeletons

To verify the linear time character of our algorithm, we first computed the timings for a number of 2D images of size

Fig. 5. Volume renderings of binary horse data ($128 \times 280 \times 232$ voxels) and its skeletons (side, front, and top views). First column: Input data. Second column: IMA-CP skeleton with $\gamma = 7$. Third column: IMA-SRP skeleton. In the skeleton renderings, the input volume is rendered partially transparent.

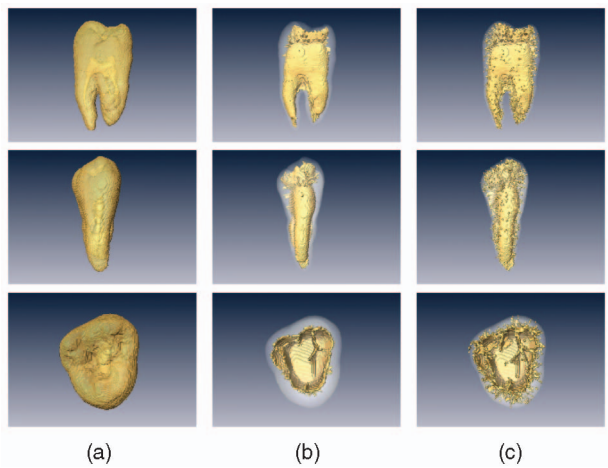

Fig. 6. Volume renderings of binary tooth data ($256 \times 256 \times 161$ voxels) and its skeletons (side, front, and top views). First column: Input data. Second column: IMA-CP skeleton with $\gamma = 7$. Third column: IMA-SRP skeleton. In the skeleton renderings, the input volume is rendered partially transparent.

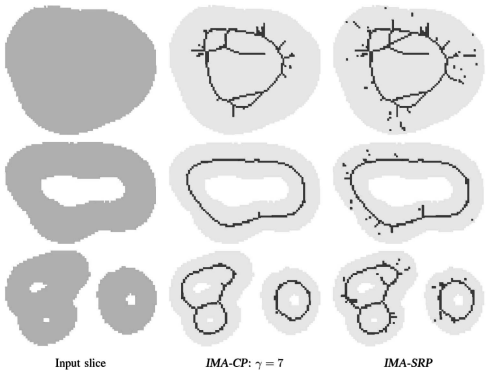

Fig. 7. Slices through skeletons of binary tooth data. First column: Input slice. Second column: IMA-CP skeleton with $\gamma = 7$. Third column: IMA-SRP skeleton. In the skeleton renderings, the input slice is also rendered.

$N \times N$, where N did not exceed 200, so that the cache size is large enough to hold all (temporary) data. The result is plotted in Fig. 8. Displayed is the cumulative CPU time (sum of system and user time, excluding data I/O) for 100 runs, both for the feature transform and the optimized skeletonization (unpruned). The images were rescaled versions of a single input image, so the fraction of foreground points was constant. We found that the timings for the feature transform step and nonoptimized skeletonization step (not shown) only very weakly depend on image content. For the optimized skeletonization step, this is evidently not the case: Since only foreground points have to be checked, this step will be dependent on the number of foreground points in the input. As can be observed in Fig. 8, both the feature transform and the skeletonization scale linearly with input size. That the optimized skeletonization scales linearly in this case can be explained by the fact that all input images were rescaled versions of the same initial image.

6.3.2 3D Skeletons

Next, we report on timing experiments for the 3D data sets listed in Section 6. The results are given in Table 1 for the feature transform and the nonoptimized and the optimized skeletonization steps. Times are averages over 10 repetitions. The input data comprised between 2 and 66 Mbytes of memory. From the table, one may infer that the behavior of the feature transform and the nonoptimized skeletonization

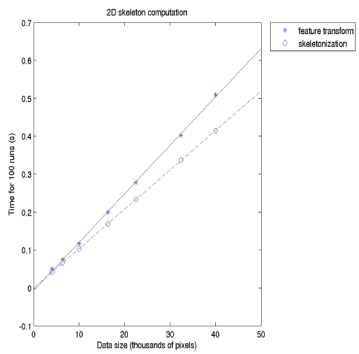

Fig. 8. Linear fit of timing results of IMA skeletonization for 2D images of size $N \times N$, where $N =$ 64, 80, 100, 128, 150, 180, 200.

TABLE 2
Timing Results for the Optimized Skeletonization Step

Data	*CP-1*	*CP-10*	*LP-1*	*SRP*
engine	0.51	0.55	0.56	0.58
vessels	0.24	0.27	0.25	0.28
tooth	0.13	0.13	0.13	0.13
horse-1	0.09	0.10	0.11	0.12
horse-2	0.19	0.21	0.21	0.22
horse-3	0.75	0.83	0.84	0.88
horse-4	1.49	1.63	1.66	1.71
horse-5	3.02	3.24	3.29	3.40

CPU times (seconds) are the averages over 10 runs. Pruning: constant (CP-1: $\gamma = 1$, CP-10: $\gamma = 10$), linear (LP-1: $\gamma = 1$), and square-root (SRP).

is still approximately linear until data sets become so large (for example, "house-5") that the storage requirements exceed the size of the internal memory and the system starts swapping. Timings for data sets of similar size do not differ much (compare the "vessels" and "horse-3" data sets). By contrast, the optimized skeletonization time is strongly dependent on the number of foreground pixels and also on the structure of the input. Notice, for example, that the optimized skeletonization time for the "tooth" data set is much smaller than that of the "engine" data set, although the latter has much fewer foreground points. Compared to the nonoptimized version, optimized skeletonization yields a speed gain factor varying from 5 to 25, depending on the input.

To study the dependence of skeletonization time on the type of pruning, we show in Table 2 timing results for the optimized skeletonization step for several types of pruning and pruning parameters. As can be seen from the table, the skeletonization time depends very weakly on pruning type. In the case of nonoptimized skeletonization, skeletonization time is virtually independent of pruning type (not shown). Therefore, we can conclude that it is computationally equally cheap to carry out skeletonization with or without pruning.

Our skeletonization algorithm is well suited for parallelization on a shared memory machine since the computation in orthogonal planes (rows, columns) is independent of the computation of other planes (rows, columns). In the different phases, the independent planes (rows, columns) are distributed over the processors. The phases must be separated by a *barrier*, which assures that all processors have completed a certain phase before any of them starts with the next one.

7 Summary and Conclusions

We have introduced a new skeleton definition, the integer medial axis (IMA), that is intended to look like the real medial axis of a smoothing of the background of the input data and tends to be connected when the complement of the background is connected, while still being computable in linear time. We proved a number of fundamental properties of the IMA skeleton and compared these with properties of the CMD (centers of maximal disks) skeleton. Several pruning methods for IMA skeletons were introduced (constant, linear, and square-root pruning) and their properties were studied. Constant and linear pruning

require the adjustment of a single parameter, whereas square-root pruning is parameter-free. The latter method was designed to suppress artifacts due to straight background discretization.

We presented an efficient algorithm to compute the IMA skeleton, based upon a linear time feature transform algorithm that is an extension of our previous EDT algorithm [16]. Experimental results for 2D and 3D input data were presented. For all program parts, explicit and compact pseudocode was given, which makes our algorithm straightforward to implement. If desired, our source code is available upon request.

Regarding skeletonization quality, we found that either constant pruning or square-root pruning gives quite acceptable results. Since there is no such thing as the "best" skeleton, the question of which skeleton one prefers depends on the criteria that one imposes upon the constructed skeleton. The main point we want to make here is that no matter what type of pruning one chooses, with whatever value of γ, our algorithm is able to compute the corresponding skeleton very fast in a time which is linear in the image size and is almost independent of pruning method.

The computation time of the feature transform and skeletonization steps was experimentally shown to follow the predicted linear-time behavior to a very good approximation. For large data sets, caching effects cause deviations from the expected linear behavior. For data of up to, say, 20 million input points, the computation time is on the order of seconds on state-of-the-art PCs. This allows our skeletonization method to be used in interactive settings, where the user can simultaneously skeletonize and visualize 3D data. For even larger data, one can use parallel versions of our algorithm (which is very easy to parallelize) or use special graphics hardware computation to keep computation time within the interactive range [21].

The CMD skeleton, which served as our prime motivation for the present work, has comparable quality to our IMA skeleton. However, it is much more expensive to compute since CMD computation is quadratic in the number of input points [14] and, hence, becomes prohibitively slow for large 3D data sets. A more efficient algorithm (probably no more than $O(n^{1.5})$) is given in [22], but its complexity is still far from linear. The unpruned IMA skeleton has many more points than CMD, but our pruned IMA-SRP skeleton has slightly fewer points than CMD and, depending on the input, fewer artifacts. In contrast, CMD is better for compression of binary image or volume data in the sense that the data are still reconstructible from CMD. Another difference with CMD is that the unpruned IMA may be connected, although we have no proof of this conjecture at the moment.

We mention several ideas for future work. First, there is the question of the topological characteristics of the (unpruned) IMA skeleton. Although we conjectured that it is connected, we thus far have no proof. Second, the issue of reconstruction from IMA-like skeletons will be considered. Finally, instead of surface skeletons, it is also interesting to look at the line skeleton (or centerline) in 3D. An algorithm based on the fast marching method was presented in [29], having complexity $\mathcal{O}(N \log N)$, with N the number of input points. An interesting question is whether the definition of IMA can be adapted to compute such a line skeleton in linear time.

REFERENCES

[1] J.L. Pfaltz and A. Rosenfeld, "Computer Representation of Planar Regions by Their Skeletons," *Comm. ACM,* vol. 10, pp. 119-125, 1972.

[2] A. Webb, *Statistical Pattern Recognition.* Arnold, 1999.

[3] J.R. Parker, *Algorithms for Image Processing and Computer Vision.* John Wiley & Sons, 1996.

[4] S. Beucher, "Digital Skeletons in Euclidean and Geodesic Spaces," *Signal Processing,* vol. 38, pp. 127-141, 1994.

[5] P. Maragos and R.W. Schafer, "Morphological Skeleton Representation and Coding of Binary Images," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 34, pp. 1228-1244, 1986.

[6] F. Meyer, "The Binary Skeleton in Three Steps," *Proc. IEEE Workshop Computer Architecture and Image Database Management,* pp. 477-483, 1985.

[7] J. Serra, *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[8] H. Talbot and L. Vincent, "Euclidean Skeletons and Conditional Bisectors," *Proc. SPIE Visual Comm. and Image Processing,* pp. 862-876, Nov. 1992.

[9] L. Vincent, "Efficient Computation of Various Types of Skeletons," *Proc. SPIE Symp. Medical Imaging,* pp. 297-311, Feb. 1991.

[10] F.Y. Shih and C.C. Pu, "A Skeletonization Algorithm by Maxima Tracking on Euclidean Distance Transform," *Pattern Recognition,* vol. 28, no. 3, pp. 331-341, 1995.

[11] H. Blum, "A Transformation for Extracting New Descriptors of Shape," *Proc. Symp. Models for the Perception of Speech and Visual Form,* W. Wathen-Dunn, ed., pp. 362-380, 1967.

[12] P.E. Danielsson, "Euclidean Distance Mapping," *Computer Graphics and Image Processing,* vol. 14, pp. 227-248, 1980.

[13] G. Borgefors, I. Nystrom, and G.S.D. Baja, "Computing Skeletons in Three Dimensions," *Pattern Recognition,* vol. 32, no. 7, pp. 1225-1236, 1999.

[14] Y. Ge and J. Fitzpatrick, "On the Generation of Skeletons from Discrete Euclidean Distance Maps," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, pp. 1055-1066, 1996.

[15] T. Hirata, "A Unified Linear-Time Algorithm for Computing Distance Maps," *Information Processing Letters,* vol. 58, pp. 129-133, 1996.

[16] A. Meijster, J.B.T.M. Roerdink, and W.H. Hesselink, "A General Algorithm for Computing Distance Transforms in Linear Time," *Math. Morphology and Its Applications to Image and Signal Processing,* J. Goutsias, L. Vincent, and D.S. Bloomberg, eds., pp. 331-340, Kluwer Academic, 2000.

[17] C.R. Maurer Jr., R. Qi, and V. Raghavan, "A Linear Time Algorithm for Computing the Euclidean Distance Transform in Arbitrary Dimensions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 2, pp. 265-270, Feb. 2003.

[18] C.R. Maurer Jr., V. Raghavan, and R. Qi, "A Linear Time Algorithm for Computing the Euclidean Distance Transform in Arbitrary Dimensions," *Information Processing in Medical Imaging,* pp. 358-364, 2001.

[19] W.H. Hesselink, M. Visser, and J.B.T.M. Roerdink, "Euclidean Skeletons of 3D Data Sets in Linear Time by the Integer Medial Axis Transform," *Math. Morphology: 40 Years On, Proc. Seventh Int'l Symp. Math. Morphology.* C. Ronse, L. Najman, and E. Decencière, eds., pp. 259-268, 2005.

[20] D. Attali and A. Montanvert, "Computing and Simplifying 2D and 3D Continuous Skeletons," *Computer Vision and Image Understanding,* vol. 67, no. 3, pp. 161-273, 1997.

[21] R. Strzodka and A. Telea, "Generalized Distance Transforms and Skeletons in Graphics Hardware," *Proc. Eurographics IEEE TCVG Symp. Visualization,* pp. 221-230, 2004.

[22] E. Remy and E. Thiel, "Look-Up Tables for Medial Axis on Squared Euclidean Distance Transform," *Proc. Int'l Conf. Discrete Geometry for Computer Imagery,* pp. 224-235, 2003.

[23] T.Y. Kong and A. Rosenfeld, "Digital Topology: Introduction and Survey," *Computer Vision, Graphics and Image Processing,* vol. 48, no. 3, pp. 357-393, 1989.

[24] W.H. Hesselink, "A Linear-Time Algorithm for Euclidean Feature Transform Sets," *Information Processing Letters,* vol. 102, pp. 181-186, 2007.
[25] D. Coeurjolly, "*d*-Dimensional Reverse Euclidean Distance Transformation and Euclidean Medial Axis Extraction in Optimal Time," *Proc. Int'l Conf. Discrete Geometry for Computer Imagery,* pp. 327-337, 2003.
[26] M. Couprie and R. Zrour, "Discrete Bisector Function and Euclidean Skeleton," *Discrete Geometry for Computer Imagery,* E. Andres, G. Damiand, and P. Lienhardt, eds., pp. 216-227, Springer, 2005.
[27] W. van der Kallen, "Integral Medial Axis and the Distance between Closest Points," *Regular and Chaotic Dynamics,* vol. 12, no. 6, pp. 734-743, 2007.
[28] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics (Proc. SIGGRAPH '88),* vol. 22, no. 4, pp. 65-74, 1988.
[29] A. Telea and J.J. van Wijk, "An Augmented Fast Marching Method for Computing Skeletons and Centerlines," *Proc. Eurographics—IEEE TCVG Symp. Data Visualization,* D. Ebert, P. Brunet, and I. Navazo, eds., pp. 251-ff, 2002.

Wim H. Hesselink received the MSc and PhD degrees in mathematics from the University of Utrecht, The Netherlands, in 1970 and 1975, respectively. He was with the University of Groningen, The Netherlands, where he worked on algebraic groups and Lie algebras. In 1984, he left pure mathematics and found a new challenge in computer science. In 1986-1987, he was on sabbatical leave with E.W. Dijkstra at the University of Texas at Austin. In 1992, he wrote a book on the weakest preconditions of recursive procedures, possibly with unbounded nondeterminacy. Since 1994, he has been the chair for Program Correctness in the Department of Computing Science at the University of Groningen. His current research concentrates on the design of concurrent algorithms, if necessary with verification by means of a mechanical theorem prover. He is a affiliate member of the IEEE.

Jos B.T.M. Roerdink received the MSc degree in theoretical physics from the University of Nijmegen, The Netherlands, in 1979 and the PhD degree in stochastic processes from the University of Utrecht, The Netherlands, in 1983. From 1983 to 1985, he was a postdoctoral fellow in stochastic processes at the University of California, San Diego. From 1986 to 1992, he was with the Centre for Mathematics and Computer Science, Amsterdam, where he worked on image processing and tomographic reconstruction. He was appointed an associate professor in 1992 and a full professor in 2003, respectively, of the Institute for Mathematics and Computing Science at the University of Groningen, The Netherlands, where he currently holds a chair in scientific visualization and computer graphics. His current research interests include biomedical visualization, neuroimaging, and bioinformatics. He is a senior member of the IEEE.