Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 23 (2005) 167–175
image AND vision COMPUTING
www.elsevier.com/locate/imavis

Exact medial axis with euclidean distance

E. Remy[a,1,*], E. Thiel[b,2]

[a]LSIS (UMR CNRS 6168)-ESIL, Case 925, 163 Av. de Luminy, 13288 Marseille, Cedex 9, France
[b]LIF (UMR CNRS 6166)-Case 901, 163 Av. de Luminy, 13288 Marseille, Cedex 9, France

Received 16 January 2004; received in revised form 29 April 2004; accepted 29 June 2004

Abstract

Medial Axis (MA), also known as Centres of Maximal Disks, is a useful representation of a shape for image description and analysis. MA can be computed on a distance transform, where each point is labelled to its distance to the background. Recent algorithms allow one to compute Squared Euclidean Distance Transform (SEDT) in linear time in any dimension. While these algorithms provide exact measures, the only known method to characterise MA on SEDT, using local tests and Look-Up Tables (LUT), is limited to 2D and small distance values [Borgefors, et al., Seventh Scandinavian Conference on Image Analysis, 1991]. We have proposed [Remy, et al., Pat. Rec. Lett. 23 (2002) 649] an algorithm which computes the LUT and the neighbourhood to be tested in the case of chamfer distances. In this article, we adapt our algorithm for SEDT in arbitrary dimension and show that results have completely different properties.

Keywords: Medial axis; Centres of maximal disks; Look-up tables; Squared euclidean distance transform; Digital shape representation

1. Introduction

Blum [2] proposed the medial axis transform (MAT), which consists in detecting the centres of the maximal disks in a 2D binary shape. Following Pfaltz and Rosenfeld [12], a disk is said to be *maximal* in a shape $\mathcal{S}$, if it is not completely covered by any single other disk in $\mathcal{S}$. The *medial axis* MA of $\mathcal{S}$ is the set of centres and radii of maximal disks in $\mathcal{S}$; an example is given in Fig. 1. Pfaltz and Rosenfeld have shown that the union of maximal disks in $\mathcal{S}$ is a covering, thus MA is a reversible coding of $\mathcal{S}$.

MA is a global representation, centred in $\mathcal{S}$, allowing shape description, analysis, simplification or compression. While MA is often disconnected and not thin in $\mathbb{Z}^n$, further treatments are applied to achieve shape analysis. In this way, MA is an important step for weighted skeleton computation [20]. A maximal disk can be included in the union of other maximal disks; so the covering by maximal disks, which is unique by construction, is not always minimal. Minimising this set while preserving reversibility can be interesting for compression, see Refs. [4,6,11].

One attractive solution to detect MA is to use a *distance transform*, denoted DT. In a distance transform on $\mathcal{S}$, each pixel is labelled with its distance to the background; it is also the radius of the largest disk in $\mathcal{S}$, centred on the pixel. A *reverse distance transform* (RDT) allow recovering the initial shape from MA.

Rosenfeld and Pfaltz have shown in Ref. [17] for the city block and chessboard distances d_4 and d_8, that it is sufficient to detect the local maxima on the DT image. For chamfer (i.e. weighted) distances using 3×3 masks, Arcelli and Sanniti di Baja [1] proved that some labels have to be lowered on the DT before identifying the local maxima; but their solution cannot be extended to larger masks. Borgefors [3] presented a method to extract MA in the case of a 5×5 chamfer mask (namely, $\langle 5,7,11\rangle$), using a look-up table (LUT). Borgefors, Ragnemalm and Sanniti di Baja had previously used the same method for SEDT in Ref. [5], but giving a partial LUT, which cannot be used for radius greater than $\sqrt{80}$.

The principle of the LUT is general: it gives for each radius value read in the DT, the minimum value of

* Corresponding author.
E-mail addresses: eric.remy@up.univ-mrs.fr (E. Remy), edouard.thiel@lim.univ-mrs.fr (E. Thiel).
[1] http://www.iut-arles.up.univ-mrs.fr/eremy/
[2] http://www.lim.univ-mrs.fr/~thiel/

0262-8856/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2004.06.007

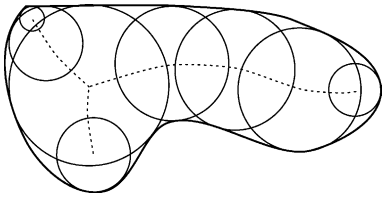

Fig. 1. Medial axis with circles.

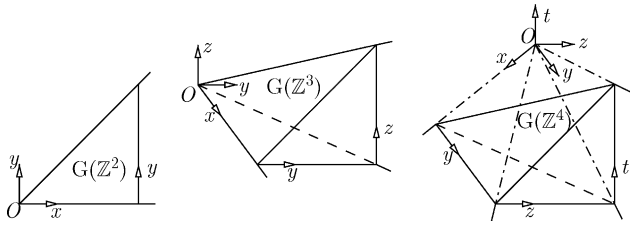

Fig. 2. The generators $G(\mathbb{Z}^n)$ for $n=2$, 3 and 4 in projection.

the neighbours that forbids a point to be in MA. The problem is to systematically compute the LUT associated with a distance function, for any radius, and also to compute the test neighbourhood (which is not necessarily 3×3 as seen later). In Ref. [15] we have shown an efficient algorithm which computes both of them for any chamfer norm in any dimension.

The first Euclidean distance transforms (EDT), proposed by Danielsson [7] and Ragnemalm [13], give approximate results, which where improved afterwards by many authors. Saito and Toriwaki [18] have presented an efficient algorithm computing exact SEDT (S for Squared) in arbitrary dimension. Recently, Hirata [9] and Meijster et al. [10] have optimised this algorithm to linear time complexity in the number of pixels. Reverse SEDT are presented in Refs. [6,19].

These exact and fast transforms bring about renewed interest in MA computation for Euclidean distance. We present in this article (which is an extended version of Ref. [16]) an adaptation of Ref. [15], which efficiently computes the LUT for SEDT in any dimension. Our algorithm also computes the test neighbourhood, and certifies that this neighbourhood is sufficient up to a given radius. We recall in Section 2 some basic notions and definitions. We present and justify in Section 3 our method. Results are given in Section 4 in the 2D and 3D cases, and we finally conclude in Section 5.

2. Definitions

2.1. Generator and grid symmetries

The rectilinear grid of $\mathbb{Z}^n$ has a number of natural symmetries, which we employ to simplify our study. We denote $\mathcal{S}_G(n)$, the group of axial and diagonal symmetries in $\mathbb{Z}^n$. The cardinal of the group is $\sharp\mathcal{S}_G(n)=2^n n!$ (which is 8, 48 and 384 for $n=2$, 3 and 4). A subset X of $\mathbb{Z}^n$ is said to be G-symmetrical if for all $\sigma\in\mathcal{S}_G(n)$ we have $\sigma(X)=X$. We call *generator* of X the subset

$$G(X)=\{(x_1,\ldots,x_n)\in X : 0\le x_n\le x_{n-1}\le\ldots\le x_1\}. \quad (1)$$

If X is G-symmetrical, the subset $G(X)$ is sufficient to reconstruct X with the G-symmetries. Fig. 2 shows $G(\mathbb{Z}^n)$ for $n=2$ (an octant), $n=3$ and 4 (cones).

2.2. Balls and reverse balls

We call *direct ball* B and *reverse ball* B^{-1} of centre $p\in\mathbb{Z}^n$ and radius $r\in\mathbb{N}$, the G-symmetric sets of points

$$B(p,r)=\{q\in\mathbb{Z}^n : d_E^2(p,q)\le r\} \quad (2)$$

$$B^{-1}(p,r)=\{q\in\mathbb{Z}^n : r-d_E^2(p,q)>0\}. \quad (3)$$

Since d_E^2 is an integer function, balls and reverse balls are linked by the relation

$$B(p,r)=B^{-1}(p,r+1). \quad (4)$$

We point out that on DT, the value DT[p] for any shape point p is the radius of the greatest *reverse* ball centred in p inside the shape, namely $B^{-1}(p,\mathrm{DT}[p])$.

2.3. Look-up tables

In the following, we denote $\mathcal{M}_{\mathrm{Lut}}$ a G-symmetric set of vectors, $\mathcal{M}^g_{\mathrm{Lut}}=G(\mathcal{M}_{\mathrm{Lut}})$ and $\vec{v}^{\,g}=G(\vec{v})$ for any vector $\vec{v}\in\mathcal{M}_{\mathrm{Lut}}$.

A shape point p is the centre of a maximal disk if there is no other shape point q such that the ball $B^{-1}(q,\mathrm{DT}[q])$ entirely covers the ball $B^{-1}(p,\mathrm{DT}[p])$. The presence of q *forbids* p to be a MA point. Suppose that it is sufficient to search q in a local neighbourhood $\mathcal{M}_{\mathrm{Lut}}$ of p. Suppose also that we know for each DT[p] the minimal value DT[q], stored in a look-up table Lut, which forbids p in direction $\vec{v}=\vec{\mathrm{pq}}$. The minimal value for p and $\vec{v}$ is stored in the array entry Lut[$\vec{v}$][DT[p]]. Because of the G-symmetry, it is sufficient to store only the relative values to $\mathcal{M}^g_{\mathrm{Lut}}$; hence the minimal value for p and $\vec{v}$ is accessed using Lut[$\vec{v}^{\,g}$][DT[p]]. Finally we have the following criterion:

$$p\in\mathrm{MA}\Leftrightarrow\mathrm{DT}[p+\vec{v}]<\mathrm{Lut}[\vec{v}^{\,g}][\mathrm{DT}[p]],\quad\forall\vec{v}\in\mathcal{M}_{\mathrm{Lut}}. \quad (5)$$

3. Computation of Lut and $\mathcal{M}_{\mathrm{Lut}}$ for SEDT

3.1. Computing an entry of Lut

The computation of an entry Lut[$\vec{v}$][r] in the LUT for $r=\mathrm{DT}[p]$ in direction $\vec{v}$, consists in finding the smallest radius R of a ball $B^{-1}(p+\vec{v},R)$ which completely

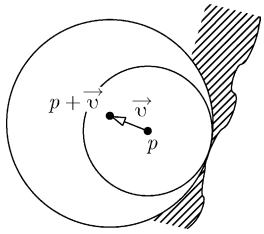

Fig. 3. Balls inside the shape.

covers $B^{-1}(p,r)$ (see Fig. 3). Since all considered balls are convex, G-symmetric and ascending by inclusion (i.e. such that if $r_1 \leq r_2$ then $B(O,r_1) \subseteq B(O,r_2)$), we can limit the covering test by restricting the two balls to $G(\mathbb{Z}^n)$. One can find R, as illustrated in Fig. 4, by decreasing the radius R_+ while keeping the ball $B^{-1}(q,R_+)$ covering the ball $B^{-1}(p,r)$, where $q = p + \vec{v} = p - \vec{v}^g$ by symmetry. A basic method, using a reverse SEDT for each step, would be prohibitive. We avoid it by using relation (4), and another distance image denoted CT^g, resulting from the cone transform in Fig. 5, where each point of $G(\mathbb{Z}^n)$ is labelled with its distance to the origin (see example Fig. 6a).

The covering of the ball $B^{-1}(q,R_+)$ over $B^{-1}(p,r)$ can be tested by simply scanning CT^g; moreover, the smallest radius R can be read in CT^g during the scan. We propose to translate both $B^{-1}(p,r)$ and $B^{-1}(q,R)$ to the origin as shown in Fig. 7. We scan each point p_1 of $G(B^{-1}(O,r))$, which by translation of vector $\vec{v}^g$ gives p_2. Values $d_E^2(O,p_1)$ and $d_E^2(O,p_2)$ are read in CT^g. We have

$$R = \max\{d_E^2(O,p_2) : p_2 = p_1 + \vec{v}^g, p_1 \in G(B^{-1}(O,r))\}, \text{so} \quad (6)$$

$$R = \max\{d_E^2(O,p_1 + \vec{v}^g) : p_1 \in G(B^{-1}(O,r))\}. \quad (7)$$

This process can be efficiently implemented (see Fig. 8), because all the covering relations (r,R) in a direction $\vec{v}^g$ can be detected during the same scan (lines 2–7). Let $\vec{v}^g_{x1}$ be the x_1 component of $\vec{v}^g$, and L be the side length of the CT^g image (chosen sufficiently large, see Section 4.2). To remain in the bounds of CT^g, the x_1 scan is limited to $L - \vec{v}^g_{x1} - 1$. For each point p_1, we look for the corresponding radius r_1 which is $\mathrm{CT}^g[p_1]+1$ by Eq. (4). Then we look

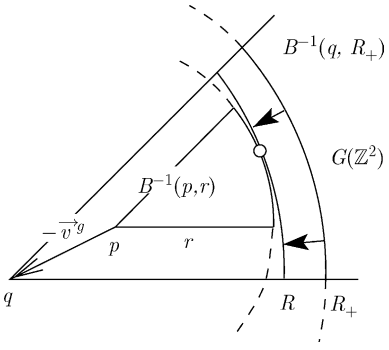

Fig. 4. Covering test on two balls restricted to $G(\mathbb{Z}^2)$.

```
Procedure CompCTg (L, CT^g) ;
1  for x_1 = 0 to L − 1, for x_2 = 0 to x_1, ..., for x_n = 0 to x_{n−1} do
2    CT^g[x_1, ..., x_n] = x_1^2 + ··· + x_n^2 ;
```

Fig. 5. Fast cone distance transform. Input: L the side length. Output: CT^g the L^n distance image to the origin for d_E^2.

or the radius r_2 of the ball passing via the point p_2. Its value is $\mathrm{CT}^g[p_2]+1 = \mathrm{CT}^g[p_1+\vec{v}^g]+1$, by Eq. (4). During the scan, we keep in $\mathrm{Lut}[\vec{v}^g][r_1]$ the greatest value found for r_2, which at the end, is R by Eq. (7).

At this stage, our algorithm gives a set of *local* covering relations, which stands for a partial ordering on the covering of balls. This ordering is not total since one can observe in Lut, cases where $r_a < r_b$ while $\mathrm{Lut}[\vec{v}^g][r_a] > \mathrm{Lut}[\vec{v}^g][r_b]$; it means that the ball covering $B^{-1}(O,r_a)$ is bigger than the ball covering $B^{-1}(O,r_b)$, which is impossible. Thus, we correct the table by assuming that in this case, $\mathrm{Lut}[\vec{v}^g][r_b]$ should at least equal $\mathrm{Lut}[\vec{v}^g][r_a]$, building this way a compatible total order (Fig. 8, lines 8–10).

3.2. Computing $\mathcal{M}_{Lut}$

Let us assume that a given $\mathcal{M}^g_{\mathrm{Lut}}$ is sufficient to extract correctly the MA from any DT which values do not exceed R_{Known}. This means that $\mathcal{M}^g_{\mathrm{Lut}}$ enables to extract, from any ball $B(O,R)$ where $R \leq R_{\mathrm{Known}}$, a MA which is by definition, the sole point O. At the beginning, $\mathcal{M}^g_{\mathrm{Lut}}$ is empty and $R_{\mathrm{Known}} = 0$.

So as to increase R_{Known} to a given R_{Target}, we propose to test (lines 3–17, Fig. 9) each ball $B(O,R)$, where $R > R_{\mathrm{Known}}$, each time extracting its DT (line 9) and then its MA (line 11), until whether R reaches R_{Target}, or a point different from O is detected in the MA of $B(O,R)$. If R reaches R_{Target}, then we know that $\mathcal{M}^g_{\mathrm{Lut}}$ enables one to extract the MA correctly, for any DT containing values lower or equal to R_{Target}. Thus, this value R_{Target} must be kept as the new R_{Known}.

On the contrary, if one extra point p is found in MA during the scan (line 11), then $\mathcal{M}^g_{\mathrm{Lut}}$ is not sufficient to properly extract the MA, since by construction $B(O,R)$ covers $B^{-1}(p,\mathrm{DT}^g[p])$. In this case we add a new vector $\vec{Op}$ in $\mathcal{M}^g_{\mathrm{Lut}}$ (line 13) and keep R for further usage, see Section 4.2. This vector is necessary and sufficient to remove p from the MA of the ball $B(O,R)$ because the current $\mathcal{M}^g_{\mathrm{Lut}}$ is validated until $R-1$; thus it enables to find all the direct balls covering $B^{-1}(p,\mathrm{DT}^g[p])$ of radii lower or equal to $R-1$. Hence, the only direct ball which is not tested is the only ball of radius R, namely $B(O,R)$. This ball is in direction $\vec{pO}$ from p and must be searched by $\mathcal{M}^g_{\mathrm{Lut}}$ to remove p. Since $\mathcal{M}_{\mathrm{Lut}}$ is G-symmetric, $B(O,R)$ is detected by adding $\vec{Op}$ in its generator.

After having added the vector, we compute the corresponding new column in Lut (line 14). Then, we ensure that this new $\mathcal{M}_{\mathrm{Lut}}$ is sufficient to remove p (line 15). This is actually a consistency test of the Lut column computation algorithm of Fig. 8, because we are sure that the new $\mathcal{M}_{\mathrm{Lut}}$ is correct.

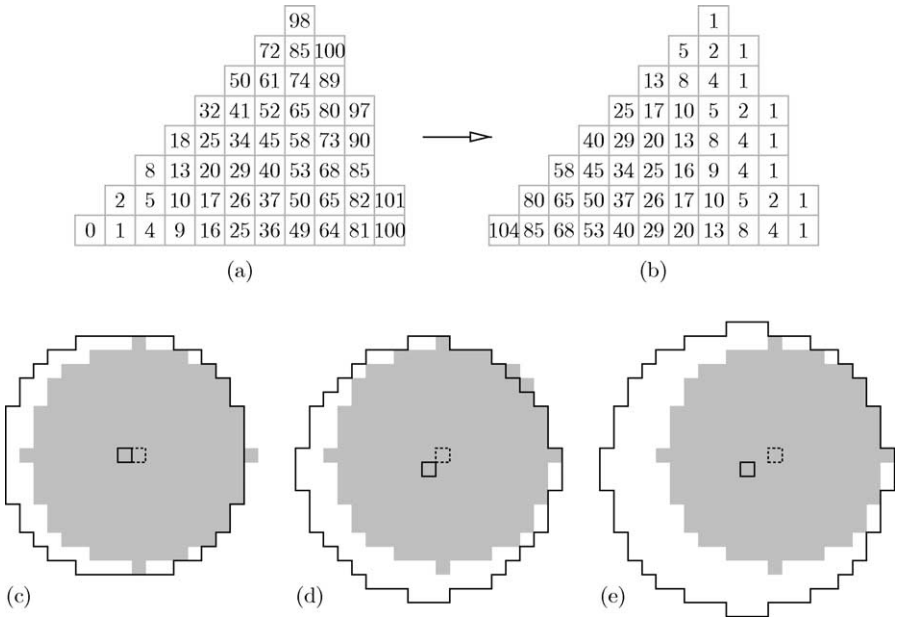

Fig. 6. Appearance of vector (2, 1) in $\mathcal{M}_{\text{Lut}}$ for $R=101$ in $\mathbb{Z}^2$. In (a), $B(101)$ is obtained using points ≤ 101 from CT^g, and gives after SEDT, $B^{-1}(104)$ in (b), on which MA is extracted. In (c), $B^{-1}(65)$ (in gray) is not overlapped by $B^{-1}(80)$ in direction (1, 0), nor in (d) by $B^{-1}(85)$ in direction (1, 1), but is overlapped in (e) by $B^{-1}(104)$ in direction (2, 1).

Once p is removed, we resume the scan for current R. Other extra points p may be detected sequentially, each time giving a new vector and Lut column. The computation of $\mathcal{M}^g_{\text{Lut}}$ is finished when R reaches R_{Target}.

The full algorithm, presented in Fig. 9, uses an adapted version of MA extraction (see Fig. 10), working on $G(\mathbb{Z}^n)$ with $\mathcal{M}^g_{\text{Lut}}$ in a single scan. Note also that the computation of DT^g (function CompSEDTg called Fig. 9, line 9), using a slightly modified SEDT working in $G(\mathbb{Z}^n)$, is mandatory, since the MA is extracted from the DT to the background. In fact, a simple threshold on image CT^g to the radius R gives only the $\text{G}(B(O,R))$ set, but not the correct DT^g labels (see Fig. 6, where values of (a) differ from (b)).

4. Results for SEDT

4.1. Computing costs

While the function d_E^2 is not a metric (triangular inequality is not satisfied), its balls respect sufficient conditions for the validity of our method (convexity, G-symmetry and ascending by inclusion). The same can be applied for discrete functions round (d_E), $\lfloor d_E \rfloor$ and $\lceil d_E \rceil$ (successfully tested).

For CompSEDTg (not presented), we have chosen to use a modified version of the algorithm in Ref. [18], which provides exact results and can be relatively easily adapted to $G(\mathbb{Z}^n)$. In particular, backward scans can be suppressed [14, Section 6.5.2]. Note that SEDT on a ball is the worst case for the complexity of Ref. [18], and that optimised algorithms [9,10] are noticeably more efficient for large radii.

The complexity in $\mathbb{Z}^n$ of CompSEDTg for a ball of radius R is $O(nR^n)$ with Refs. [9,10] or $O(nR^{n+1})$ with Ref. [18]. The complexity of CompLutCol is $O(R^n)$ (one scan of $G(\mathbb{Z}^n)$ plus one scan of a Lut column). The complexity of IsMAg, with a number k of directions to test, is $O(kR^n)$ in the worst case, that is to say, when p is detected as a MA point. Since this event is seldom, the algorithm returns almost always early, hence the real cost of IsMAg is negligible. In CompLutMask, the complexity of one iteration of the main loop (lines 4–16 in Fig. 9) is thus the complexity of

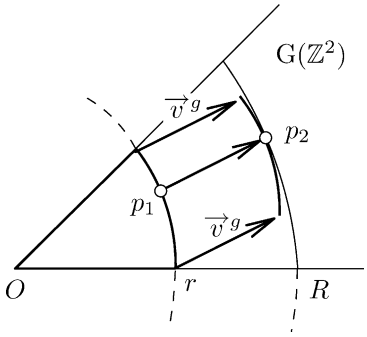

Fig. 7. Translated covering test on CT^g.

```
    Procedure CompLutCol (CT^g, L, v^g, R_max, Lut[v^g]) ;
1   for r = 0 to R_max do Lut[v^g][r] = 0 ;        // Initialize Lut[v^g] to 0.
2   for x_1 = 0 to L − v^g_{x_1} − 1, for x_2 = 0 to x_1, ..., for x_n = 0 to x_{n−1} do
3    {
4     r_1 = CT^g[x_1, ..., x_n] + 1 ;     // Radius of the ball where p_1 is located,
5     r_2 = CT^g[(x_1, ..., x_n) + v^g] + 1 ;                    // same for p_2.
6     if r_1 ≤ R_max and r_2 > Lut[v^g][r_1] then Lut[v^g][r_1] = r_2;
7    }
8   r_b = 0 ;
9   for r_a = 0 to R_max do
10    if Lut[v^g][r_a] > r_b then r_b = Lut[v^g][r_a] else Lut[v^g][r_a] = r_b ;
```

Fig. 8. Lut column computation. Input: CT^g the cone, L the side length, $\vec{v}^g$ the direction of the search, $R_{\max}$ the greatest radius value to be verified in Lut. Output: the column $\text{Lut}[\vec{v}^g]$ is filled with the correct values.

```
Procedure CompLutMask (L, M^g_Lut, R_Known, R_Target, Lut) ;
1  CompCTg (L, CT^g) ;
2  for each v^g in M^g_Lut do CompLutCol(CT^g, L, v^g, R_Target, Lut[v^g]) ;
3  for R = R_Known + 1 to R_Target do
4   {
5    for x_1 = 0 to L − 1, for x_2 = 0 to x_1, ..., for x_n = 0 to x_{n−1} do
6      if CT^g[x_1, ..., x_n] ≤ R
7        then DT^g[x_1, ..., x_n] = 1
8        else DT^g[x_1, ..., x_n] = 0 ;          // Copy G(B(R)) to DT^g
9    CompSEDTg (L, DT^g) ;
10   for x_1 = 1 to L − 1, for x_2 = 0 to x_1, ..., for x_n = 0 to x_{n−1} do
11     if DT^g[x_1, ..., x_n] ≠ 0 and IsMAg((x_1, ..., x_n), M^g_Lut, Lut, DT^g) then
12      {
13       M^g_Lut = M^g_Lut ∪ (x_1, ..., x_n ; R) ;          // Insert the new vector
14       CompLutCol (CT^g, L, (x_1, ..., x_n), R_Target, Lut[x_1, ..., x_n]) ;
15       if IsMAg ((x_1, ..., x_n), M^g_Lut, Lut, DT^g) then error ;
16      }
17  }
```

Fig. 9. Full $\mathcal{M}^g_{\text{Lut}}$ and Lut Computation. Input: L the side length, $\mathcal{M}^g_{\text{Lut}}$, R_{Known} and R_{Target}. Output: Lut, $\mathcal{M}^g_{\text{Lut}}$ and R_{Target}. At first call, $\mathcal{M}^g_{\text{Lut}}$ and R_{Known} must be set to $\varnothing$ and 0, respectively. After exit, R_{Known} must be set to R_{Target}.

CompSEDTg. As CompLutMask makes radius R increase, its total cost grows quite fast.

We present the results of our method in 2D and 3D in Figs. 11 and 12. Computing the $\mathcal{M}^g_{\text{Lut}}$ shown Fig. 11 takes 590 s, while computing one corresponding Lut column takes 0.004 s, for $L=400$ and from $R_{\text{Known}}=0$ to $R_{\text{Target}}=128\,200$ (on a Pentium 4 at 2.26 GHz with Debian Gnu/Linux 2.4.19). This load is explained by the systematic test of about 26 000 balls. As expected, CompLutCol is very fast, whereas CompLutMask is much slower, and its resulting (and compact) $\mathcal{M}^g_{\text{Lut}}$ should thus be saved for further re-usage.

The memory required to store Lut is mRe, where m is the number of columns in $\mathcal{M}^g_{\text{Lut}}$ for R, and e is the size of one long integer (to store d^2_E values). Since R grows with the square of the radius in pixel of the largest Euclidean ball tested, the memory cost of Lut becomes important for large images (for instance the size of the Lut corresponding to Fig. 11 is 23 MB). In Fig. 13 we have shown the behavior of the number m of vectors in $\mathcal{M}^g_{\text{Lut}}$ according to the appearance radius in pixels $\sqrt{R}$, up to $\sqrt{3036452}\approx 1742$ in 2D (left) and $\sqrt{16384}=128$ in 3D (right).

Memory can be saved by storing only possible values of d^2_E. The set of possible values in nD is $S=\{x_1^2+\cdots+x_n^2\le R : x_i\in[0..R]\}$. The Lut entries are then accessed by Lut[$\vec{v}^g$][index[r]], where *index* is a table of size $R+1$, built in a single scan on CT^g, which gives for any $r\in[0\ldots R]$ the rank index[r] in S. To express the gain of Lut space in nD we introduce the fraction $\delta(n,r)$ of numbers smaller or equal to r that are expressible as the sum of n squares. Since any positive integer can be decomposed into sum of four (or more) squares (Lagrange's thm., see [8, Section 20.5]), we have $\delta(n,r)=1$ for $n\ge 4$. Lagrange also stated that

```
Function IsMAg (p, M^g_Lut, Lut, DT^g) ;
1  for each v^g in M^g_Lut do
2    if p − v^g ∈ G(Z^n) then                    // Test only in G(Z^n).
3      if DT^g[p − v^g] ≥ Lut[v^g][DT^g[p]] then return false ;
4  return true ;
```

Fig. 10. Fast extraction of MA points from $G(B)$. Input: p the point to test, $\mathcal{M}^g_{\text{Lut}}$ the generator of the Lut neighbourhood, Lut the look-up table, DT^g the distance transform of the section of the ball. Output: returns true if point p is detected as MA in DT^g.

i	x,y	R	i	x,y	R
1	1,0	1	24	8,7	28 564
2	1,1	2	25	7,6	29 042
3	2,1	101	26	9,4	35 113
4	3,1	146	27	8,3	38 900
5	3,2	424	28	9,5	44 433
6	4,1	848	29	11,6	44 433
7	5,1	1 370	30	8,5	46 660
8	6,1	2 404	31	14,1	57 128
9	4,3	3 049	32	12,1	58 084
10	7,1	3 250	33	11,4	59 417
11	5,2	3 257	34	9,8	64 089
12	7,5	3 700	35	13,1	64 528
13	5,3	4 709	36	15,1	66 050
14	7,3	5 954	37	12,7	66 820
15	5,4	9 805	38	10,3	72 194
16	8,1	11 237	39	13,7	75 242
17	9,1	11 889	40	11,3	76 040
18	10,1	14 885	41	11,5	91 012
19	7,4	19 465	42	9,7	92 240
20	11,1	20 738	43	13,6	104 452
21	6,5	22 261	44	11,2	109 609
22	7,2	22 736	45	11,9	117 137
23	9,2	26 216	46	10,9	125 512
			47	16,1	128 178

Fig. 11. Beginning of $\mathcal{M}^g_{\text{Lut}}$ for $\mathbb{Z}^2$ (appearance rank i, coordinates, appearance radius R).

i	x,y,z	R	i	x,y,z	R
1	1,0,0	1	35	5,3,3	459
2	1,1,0	2	36	7,3,1	499
3	1,1,1	3	37	11,5,2	499
4	2,1,1	26	38	8,2,1	546
5	2,1,0	49	39	4,4,1	548
6	3,1,0	65	40	8,5,2	548
7	3,1,1	67	41	5,5,2	550
8	2,2,1	83	42	6,3,2	558
9	3,2,1	89	43	6,5,2	558
10	4,2,1	134	44	5,4,3	568
11	4,1,1	146	45	7,4,2	571
12	4,3,1	165	46	7,1,1	579
13	3,3,1	180	47	7,2,2	603
14	4,3,2	195	48	6,5,1	651
15	5,1,1	198	49	6,4,3	660
16	3,2,0	203	50	9,2,1	690
17	5,3,1	203	51	7,5,1	691
18	5,3,2	211	52	7,5,3	691
19	3,2,2	229	53	9,3,1	699
20	5,2,1	233	54	7,3,2	714
21	4,1,0	292	55	8,3,1	722
22	5,2,0	298	56	4,3,0	725
23	6,3,1	306	57	7,4,1	746
24	6,2,1	308	58	8,4,1	770
25	5,2,2	317	59	6,5,3	780
26	5,1,0	325	60	8,5,1	819
27	5,4,1	341	61	8,6,3	824
28	7,2,1	341	62	8,4,3	841
29	3,3,2	355	63	9,6,2	859
30	6,4,1	373	64	8,5,3	867
31	7,5,2	373	65	7,6,4	875
32	6,1,1	402	66	8,1,1	902
33	5,4,2	405	67	9,5,3	915
34	5,3,0	425	68	10,3,1	931
			69	5,5,3	947

Fig. 12. Beginning of $\mathcal{M}^g_{\text{Lut}}$ for $\mathbb{Z}^3$ (appearance rank i, coordinates, appearance radius R).

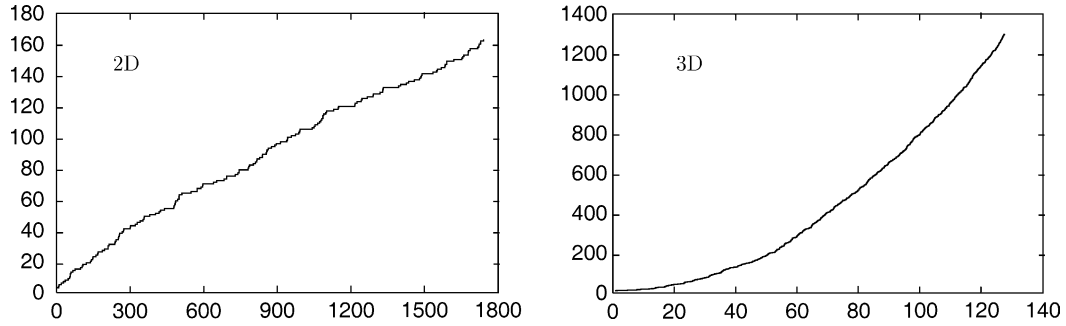

Fig. 13. Behavior of the number m of vectors in $\mathcal{M}^g_{\mathrm{Lut}}$ according to the appearance radius in pixels $\sqrt{R}$ in 2D (left) and in 3D (right).

numbers expressible as the sum of three squares are those not of the form $4^a(8^b+7)$ for $a,b\geq 0$. (see for instance [22]); therefore $\lim_{r\to\infty}\delta(3,r)=1-\left(\frac{1}{4^0}+\frac{1}{4^1}+\frac{1}{4^2}+\cdots\right)\cdot\frac{1}{8}=1-\frac{4}{3}\cdot\frac{1}{8}=\frac{5}{6}$ with fast convergence. For $n=2$ we have slow convergences $\lim_{r\to\infty}\delta(2,r)=0$ and $\lim_{r\to\infty}\delta(2,r)\sqrt{\ln r}=K$, where $K\approx 0.764223653$ is known as the Landau–Ramanujan constant, see Ref. [22]. Accordingly, in 4D or more, no space can be saved by the use of such indexes. In 3D, a gain of 1/6 may be achieved. In 2D a substantial gain is obtained: for the Lut corresponding to Fig. 11, about 78% may be saved, with only 5.1 MB remaining to store.

4.2. Extracting medial axis

A sample usage of the Lut given by Fig. 14 and formula (5) is: a point valued 4 on DT is not a MA point if, following third entry in table, it has at least a (1, 0)-neighbour ≥ 6, or a (1, 1)-neighbour ≥ 9, or a (2, 1)-neighbour ≥ 14, etc. As explained above, the table only shows possible radii r.

In Figs. 11 and 12 are given the vectors of $\mathcal{M}^g_{\mathrm{Lut}}$ in 2D and 3D, respectively, and also their appearance radius R during CompLutMask. Keeping this radius is important because it allows one to limit the number of directions to test for each point during whole MA extraction. In a DT where the greatest value is $R_{\max}$, it is necessary and sufficient to take the subset $\mathcal{M}^{R_{\max}}_{\mathrm{Lut}}=\{(\vec{v};R)\in\mathcal{M}_{\mathrm{Lut}} : R<R_{\max}\}$ as the test neighbourhood to detect all MA points. In fact, CompLutMask guarantees that $\mathcal{M}^{R_{\max}}_{\mathrm{Lut}}$ is necessary and sufficient up to $R_{\mathrm{Known}}=R_{\max}-1$ in CT^g (as a radius of direct ball), thus by Eq. (4), up to $R_{\max}$ in DT (as a radius of reverse ball). For example in Fig. 11, if $R_{\max}=101$ on DT, then the test neighbourhood will be limited to (1, 0)-neighbours and (1, 1)-neighbours.

During CompLutMask, both images CT^g and DT^g need to be large enough to contain all tested direct balls, and especially the greatest one $B(R_{\max}-1)$. Therefore we use images of size L^n, where $L=\lfloor\sqrt{R_{\max}-1}\rfloor+1$ is the radius of $B(R_{\max}-1)$ in number of pixels.

The extraction of MA from a binary image I can be divided in the following steps. One must first compute SEDT, then search $R_{\max}$ in the resulting DT. Next, CompLutMask is applied using the $R_{\max}$ value as R_{Target}; this step can be avoided if a sufficient $\mathcal{M}^g_{\mathrm{Lut}}$, computed once for all, is already stored. The subset $\mathcal{M}^{R_{\max}}_{\mathrm{Lut}}$ is then used to extract MA, which is initialised to shape points. To minimise memory usage, we propose to allocate only one Lut column at a time, instead of computing for $R_{\max}$ and $\sharp\mathcal{M}^{R_{\max}}_{\mathrm{Lut}}$ the whole Lut, which might be very large as seen in Section 4.1: for each vector $\vec{v}^g$ in $\mathcal{M}^{R_{\max}}_{\mathrm{Lut}}$, we overwrite the previous column using CompLutCol, then reject from MA all the points which do not fulfill Eq. (5) with the G-symmetries of $\vec{v}^g$. This way, the MA set often decrease extremely fast at each step.

4.3. On vectors appearance in $\mathcal{M}^g_{Lut}$

Two reverse balls of radii r and r' are said *equivalent* if the sets of pixels $B^{-1}(O,r)$ and $B^{-1}(O,r')$ are the same.

r	1,0	1,1	2,1	3,1	3,2	81	98	107	126	147	158
1	2	3	6	11	14	82	101	107	126	147	158
2	5	6	11	18	21	85	102	107	126	149	158
4	6	9	14	21	26	89	105	114	131	154	165
5	10	11	18	27	30	90	107	118	137	158	171
8	11	14	21	30	35	97	110	118	138	161	171
9	14	19	26	35	42	98	117	126	147	170	181
10	17	19	27	38	42	100	117	129	147	170	182
13	18	21	30	41	46	101	122	131	150	171	186
16	21	26	35	46	53	104	123	131	150	174	186
17	26	27	38	51	54	106	126	131	154	179	186
18	27	30	41	54	59	109	126	137	158	181	194
20	27	33	42	54	62	113	131	138	161	186	195
25	30	35	46	59	66	116	131	146	165	186	203
26	37	42	53	66	75	117	138	147	170	195	206
29	38	42	54	69	75	121	138	150	171	195	209
32	41	46	59	74	81	122	145	150	171	198	209
34	42	51	62	75	86	125	146	150	174	201	209
36	46	53	66	81	90	128	149	158	181	206	219
37	50	53	66	83	90	130	149	163	182	206	222
40	51	54	69	86	91	136	154	165	186	213	226
41	54	59	74	91	98	137	158	171	194	219	234
45	54	62	75	91	101	144	161	171	195	222	234
49	59	66	81	98	107	145	170	171	198	227	234
50	65	66	83	102	107	146	171	182	203	230	245
52	66	73	86	105	114	148	171	182	206	233	246
53	66	75	90	107	118	149	174	182	206	235	246
58	69	75	91	110	118	153	174	186	209	235	251
61	74	81	98	117	126	157	179	186	213	242	251
64	75	86	101	118	131	160	181	194	219	246	261
65	82	86	102	123	131	162	186	195	222	251	262
68	83	90	107	126	137	164	186	201	222	251	266
72	86	91	110	131	138	169	186	203	226	251	270
73	86	99	114	131	146	170	197	206	233	262	275
74	91	99	117	138	147	173	198	209	234	262	278
80	91	101	118	138	150	178	201	209	235	266	278

Fig. 14. Beginning of Lut for $\mathbb{Z}^2$ (radius r, next columns Lut$[\vec{v}^g][r]$).

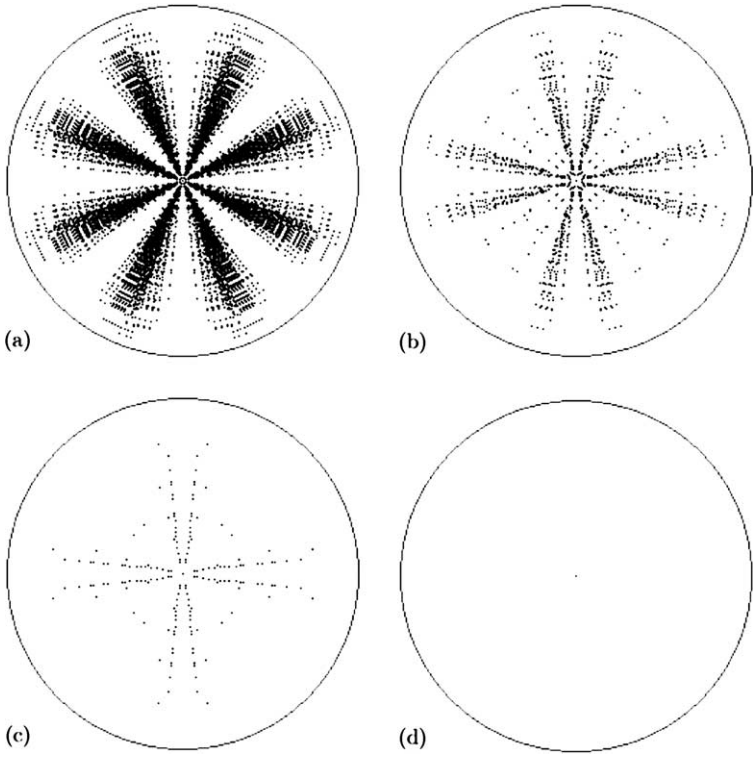

Fig. 15. Medial axis of a 2D Euclidean ball: computed with $\mathcal{M}_{\mathrm{Lut}}$ limited to (a) the 3×3 neighbourhood; (b) 5×5; (c) 7×7; (d) using $\mathcal{M}_{\mathrm{Lut}}^{R_{\max}}$.

The *equivalence class* of a reverse ball is the interval of radii for which the reverse balls are equivalent. In $\mathbb{Z}^n$, the equivalence classes are easily obtained by underlining *possible values* in DT (i.e. integers which can be written in sum of n squares); the equivalence class of a possible value b is $\{a,\ldots\underline{b}\}$ where $a-1$ is the largest possible value less than b. The first equivalence classes in 2D are $\{\underline{1}\}$, $\{\underline{2}\}$, $\{3,\underline{4}\}$, $\{\underline{5}\}$, $\{6,7,\underline{8}\}$, $\{\underline{9}\}$, $\{\underline{10}\}$, $\{11,12,\underline{13}\}$, etc.

Equivalence classes of size greater than 1 exist in 2D and 3D because the sums of two or three squares do not fill $\mathbb{N}$. All the balls are different for dimension $n \geq 4$ because of the Lagrange's theorem; we think that this might have implications over properties of $\mathcal{M}_{\mathrm{Lut}}$ and Lut which are linked to equivalence classes.

Our algorithm CompLutCol in Fig. 8 gives the lowest bound of each equivalence class. We remark that the values published in Ref. [5] correspond to the highest bounds; in that sense, the two tables must be considered as equivalent. Fig. 11 also confirms the 3×3 test neighbourhood used in Ref. [5] for radii less than 80 in 2D, because the third direction only appears for $R=101$.

We illustrate in Fig. 6 the appearance of the direction (2, 1) in $\mathcal{M}_{\mathrm{Lut}}$ for $R=101$ in $\mathbb{Z}^2$. The radius $R=101$ of a direct ball (Fig. 6a) corresponds by Eq. (4) to radius $R'=101+1$ of reverse ball. Since equivalence class of 102 is $\{102, 103, \underline{104}\}$, CompSEDTg labels O to 104 (Fig. 6b). When extracting MA with two test directions (0,1) and (1,1), the point labelled 65 is detected since its reverse ball is not completely overlapped by the reverse balls of its neighbours (Fig. 6c and d), while it is overlapped in direction (2,1) (Fig. 6e).

Extracting MA using a subset $\mathcal{N}$ of $\mathcal{M}_{\mathrm{Lut}}^{R_{\max}}$, for instance a $(2k+1)^n$ neighbourhood ($k>0$), provides a superset of MA, because a number of points, called *spurious* points, can be detected, which do satisfy Eq. (5) for $\mathcal{N}$ but not for $\mathcal{M}_{\mathrm{Lut}}^{R_{\max}}$. To illustrate the presence of spurious points, we show in Fig. 15 in 2D and Fig. 16 in 3D the MA obtained with some $(2k+1)^n$ neighbourhoods on euclidean spheres (in Fig. 15, a sphere of radius 15,000 corresponding to a diameter of 245 pixels; in Fig. 16, radius 2257 and diameter 99 pixels). By definition, their exact MA is the origin. Figs. 15 and 16 show that numerous spurious points are detected when k is small. In the 3D example, 22% of the original shape points are still present for $k=1$ (3×3×3 neighbourhood Fig. 16 (b)); 6% of shape points are still present in (c) for $k=2$, 1.7% in (d) for $k=3$ and 0.6% in (e) for $k=4$.

4.4. Conjecture

Our experiments in 2D and 3D suggest that $\mathcal{M}_{\mathrm{Lut}}$ is not bound for d_E^2, unlike chamfer distances (see Ref. [15]) Fig. 17 geometrically represents the set of vectors in $\mathcal{M}_{\mathrm{Lut}}^{g}$ from Fig. 11 with their rank of appearance. While layout seems random, one can note that all $\mathcal{M}_{\mathrm{Lut}}$ points are visible points. A point $(x_1,\ldots,x_n)$ is said *visible* (from the origin) if $\gcd(x_1,\ldots,x_n)=1$; the set of visible points in $\mathbb{Z}^n$ is denoted $\mathcal{V}^n$ (see Ref. [21]). When carrying on computation of $\mathcal{M}_{\mathrm{Lut}}^{g}$ with CompLutMask, all visible points seems to be gradually

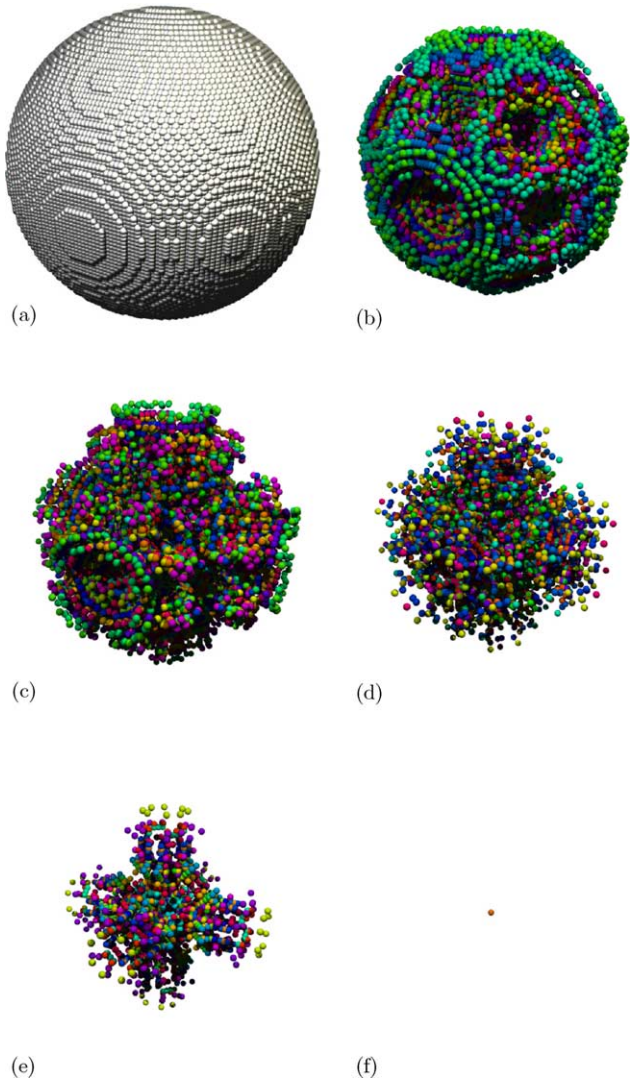

Fig. 16. Medial axis of a 3D Euclidean ball (a) computed with $\mathcal{M}_{\text{Lut}}$ limited to (b) the 3×3 neighbourhood; (c) 5×5; (d) 7×7; (e) 9×9; (f) using $\mathcal{M}_{\text{Lut}}^{R_{\max}}$.

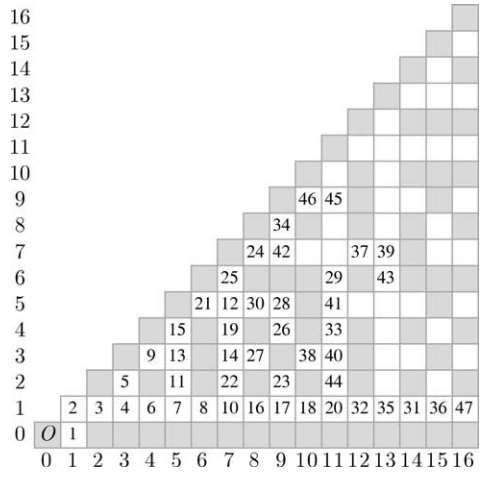

Fig. 17. Representation of $\mathcal{M}_{\text{Lut}}^{g}$ points (values show appearance ranks, white squares are visible points, grey squares non-visible).

detected, while non-visible points never are. We therefore propose the conjecture:

$$\lim_{R\to\infty} \mathcal{M}_{\text{Lut}}^{R} = \mathcal{V}^{n}. \tag{8}$$

These properties for d_E^2 are very different from those of chamfer distances (see Ref. [15]), where $\mathcal{M}_{\text{Lut}}$ are always bound, Lut are bound in most cases, and non-visible points may appear in $\mathcal{M}_{\text{Lut}}$. We think this is linked to the number of normals of the balls, which is unbound for infinite Euclidean balls, while bound for chamfer balls.

5. Conclusion

The computation of the MA from the SEDT is detailed for arbitrary dimension. The principle of MA extraction using Lut was already published for d_E^2 in 2D for small values and 3×3 neighbourhood in Ref. [5], but no general method to compute them was given. We have introduced the mask $\mathcal{M}_{\text{Lut}}$, which stores the test neighbourhood used during the MA extraction. We showed that, in the general case, the mask $\mathcal{M}_{\text{Lut}}$ is greater than just the 3^n neighbourhood. We have presented and justified efficient algorithms which compute both Lut and $\mathcal{M}_{\text{Lut}}$ for d_E^2. Our algorithms certify that $\mathcal{M}_{\text{Lut}}$ is sufficient up to a given ball radius. We give a sample Lut table in 2D for comparison with Ref. [5]. We give two sets of $\mathcal{M}_{\text{Lut}}^{g}$ in 2D and 3D, which enable a simple MA extraction using only the Lut table computation algorithm (provided that the greatest radius R in the image is lower than 128, 178 in 2D and 947 in 3D).

Our experimentations point out that, in the case of d_E^2, the neighbourhood $\mathcal{M}_{\text{Lut}}$ to test is a set of visible points. Unlike seen in the case of chamfer distances in Ref. [15], this set seems to grow forever as the radius R of the greatest possible ball in the image grows. A further work needs to be done to get a better understanding of the inclusions of discrete Euclidean balls and to find arithmetical rules.

6. Electronic appendix

Some more examples of medial axes, Lut and $\mathcal{M}_{\text{Lut}}^{g}$ in several dimensions and program sources (in C language) are available at

http://www.lim.univ-mrs.fr/∼thiel/IVC2004/

References

[1] C. Arcelli, G. Sanniti di Baja, Finding local maxima in a pseudo-Euclidean distance transform, Comp. Vis., Graph. Image Proc. 43 (1988) 361–367.

[2] H. Blum, A transformation for extracting new descriptors of shape in: W. Wathendunn (Ed.), Models for the Perception of Speech and Visual Form, MIT Press, Cambridge, 1967, pp. 362–380.

[3] G. Borgefors, Centres of maximal disks in the 5–7–11 distance transform, in: Eighth Scandinavian Conference on Image Analysis, Tromsø, Norway, 1993 pp. 105–111.
[4] G. Borgefors, I. Nyström, Efficient shape representation by minimizing the set of centres of maximal discs/spheres, Pat. Rec. Lett. 18 (1997) 465–472.
[5] G. Borgefors, I. Ragnemalm, G. Sanniti di Baja, The Euclidean Distance Transform: finding the local maxima and reconstructing the shape, in: Seventh Scandinavian Conference on Image Analysis, Aalborg, Denmark, vol. 2 1991 pp. 974–981.
[6] D. Coeurjolly, d-Dimensional Reverse Euclidean Distance Transformation and Euclidean Medial Axis Extraction in Optimal Time, in: 11th DGCI, Discrete Geometry for Computer Image, volume 2886 of Lectures Notes in Computer Science, Naples, Italy, 2003 pp. 327–337.
[7] P.E. Danielsson, Euclidean distance mapping, Comp. Graph. Image Proc. 14 (1980) 227–248.
[8] G.H. Hardy, E.M. Wright, An Introduction to the Theory of Numbers, fifth ed., Oxford University Press, Oxford, 1978.
[9] T. Hirata, A unified linear-time algorithm for computing distance maps, Inf. Proc. Lett. 58 (1996) 129–133.
[10] A. Meijster, J.B.T.M. Roerdink and W.H. Hesselink, A general algorithm for computing distance transforms in linear time, in: Mathematical Morphology and its Applications to Image and Signal processing; J. Goutsias, L. Vincent, and D. S. Bloomberg (Eds.), Kluwer, 2000, pp. 331–340.
[11] F. Nilsson, P.E. Danielsson, Finding the minimal set of maximum disks for binary objects, Graph. Models Image Proc. 59 (1) (1997) 55–60.
[12] J.L. Pfaltz, A. Rosenfeld, Computer representation of planar regions by their skeletons, Commun. ACM 10 (1967) 119–125.
[13] I. Ragnemalm, The Euclidean distance transform in arbitrary dimensions, Pat. Rec. Lett. 14 (11) (1993) 883–888.
[14] E. Remy. Normes de chanfrein et axe médian dans le volume discret, PhD Thesis, Univ. de la Méditerranée, Aix-Marseille 2, Dec 2001.
[15] E. Remy, E. Thiel, Medial axis for chamfer distances: computing look-up tables and neighbourhoods in 2D or 3D, Pat. Rec. Lett. 23 (6) (2002) 649–661.
[16] E. Remy, E. Thiel, Look-Up Tables for Medial Axis on Squared Euclidean Distance Transform, in: 11th DGCI, Discrete Geometry for Computer Image, volume 2886 of Lectures Notes in Computer Science, Naples, Italy, 2003 pp. 224–235.
[17] A. Rosenfeld, J.L. Pfaltz, Sequential operations in digital picture processing, J. ACM 13 (4) (1966) 471–494.
[18] T. Saito, J.I. Toriwaki, New algorithms for Euclidean distance trans. of an n-dim. digitized picture with applications, Pat. Rec. 27 (11) (1994) 1551–1565.
[19] T. Saito, J.I. Toriwaki, Reverse distance transformations and skeletons based upon the Euclidean metric for *n*-dimensional digital pictures, IECE Trans. Inf. Syst. E77-D (9) (1994) 1005–1016.
[20] G. Sanniti di Baja, E. Thiel, A skeletonization algorithm running on path-based distance maps, Image Vis. Comput. 14 (1) (1996) 47–57.
[21] E. Thiel, Géométrie des distances de chanfrein, Docent, Univ. de la Méditerranée, Aix-Marseille 2, Dec 2001. (http://www.lim.univ-mrs.fr/∼thiel/hdr
[22] E.W. Weisstein, Landau–Ramanujan Constant, http://mathworld.wolfram.com/Landau–RarnanujanConstant.html