Look-Up Tables for Medial Axis on Squared Euclidean Distance Transform

Eric Remy[1] and Edouard Thiel[2]

[1] LSIS (UMR CNRS 6168) - ESIL, Case 925,
163 Av. de Luminy, 13288 Marseille Cedex 9, FRANCE
Eric.Remy@up.univ-mrs.fr

[2] LIF (UMR CNRS 6166) - Case 901,
163 Av. de Luminy, 13288 Marseille Cedex 9, FRANCE
Edouard.Thiel@lim.univ-mrs.fr
http://www.lim.univ-mrs.fr/~thiel

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 to compute Squared Euclidean Distance Transform (SEDT) in linear time in any dimension. While these algorithms provide exact measures, the only known method to characterize MA on SEDT, using local tests and Look-Up Tables, is limited to 2D and small distance values [5]. We have proposed in [14] an algorithm which computes the look-up table and the neighbourhood to be tested in the case of chamfer distances. In this paper, 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 proposed in [2] the medial axis transform (MAT), which consists in detecting the centres of the maximal disks in a 2D binary shape. Following Pfaltz and Rosenfeld in [11], 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 Figure 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 [17]. 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. Minimizing this set while preserving reversibility can be interesting for compression, see [10, 4].

I. Nyström et al. (Eds.): DGCI 2003, LNCS 2886, pp. 224-235, 2003.

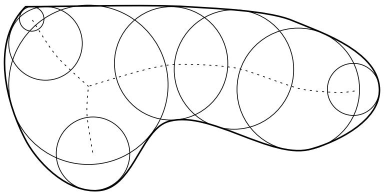

Fig. 1. Medial Axis with circles.

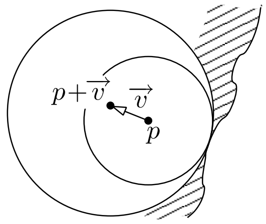

Fig. 2. Balls inside the shape.

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 to recover the initial shape from MA.

Rosenfeld and Pfaltz have shown in [15] 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 proved in [1] 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 presented in [3] 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. Borgefors, Ragnemalm and Sanniti di Baja have previously used the same method for SEDT in [5], but giving a partial look-up table, which cannot be used for radius greater than $\sqrt{80}$.

The principle of look-up table (LUT) is general: it gives for each radius value read in the DT, the minimum value of the neighbours which 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 [14] 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 [6] and Ragnemalm [12], give approximate results, which where improved afterwards by many authors. Saito and Toriwaki in [16] have presented an efficient algorithm computing exact SEDT (S for Squared) in arbitrary dimension. Recently, Hirata [8] and Meijster et al. [9] have optimized this algorithm to linear time complexity in the number of pixels. Reverse SEDT can be easily derived from [16, 8, 9].

These exact and fast transforms bring about renewed interest in MA computation for Euclidean distance. We present in this paper an adaptation of [14], 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 §2 some basic notions and definitions. We present and justify in §3 our method. Results are given in §4 in the 2D and 3D cases, and we finally conclude in §5.

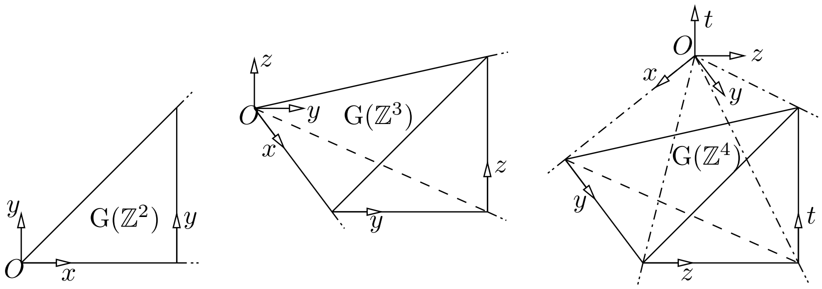

Fig. 3. The generators $\mathrm{G}(\mathbb{Z}^n)$ for $n = 2$, 3 and 4 in projection.

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 $\#\mathcal{S}_G(n) = 2^n n!$ (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

$$\mathrm{G}(X) = \left\{ (x_1, ..., x_n) \in X \;:\; 0 \leqslant x_n \leqslant x_{n-1} \leqslant \ldots \leqslant x_1 \right\} . \tag{1}$$

If X is G-symmetrical, the subset $\mathrm{G}(X)$ is sufficient to reconstruct X with the G-symmetries. Figure 3 shows $\mathrm{G}(\mathbb{Z}^n)$ 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

$$B(p, r) = \left\{ q \in \mathbb{Z}^n \;:\; d_E^2(p, q) \leq r \right\} \tag{2}$$

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

Since d_E^2 is integral, balls and reverse balls are linked by the relation

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

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 $B^{-1}(p, DT[p])$.

2.3 Look-Up Tables

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

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

$$p \in \text{MA} \quad \Longleftrightarrow \quad DT[p + \overrightarrow{v}] < Lut[\overrightarrow{v}^g][DT[p]] \ , \ \ \forall \overrightarrow{v} \in \mathcal{M}_{Lut} \ . \tag{5}$$

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

3.1 Computing an entry of *Lut*

The computation of an entry $Lut[\overrightarrow{v}][r]$ in the look-up table for $r = DT[p]$ in direction $\overrightarrow{v}$, consists in finding the smallest radius R of a ball $B^{-1}(p+\overrightarrow{v}, R)$ which completely covers $B^{-1}(p, r)$ (see Figure 2). Since all considered balls are convex, G-symmetric and 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 $\text{G}(\mathbb{Z}^n)$. One can find R, as illustrated in Figure 4, by decreasing the radius R_+ while keeping the ball $B^{-1}(q, R_+)$ covering the ball $B^{-1}(p, r)$, where $q = p + \overrightarrow{v} = p - \overrightarrow{v}^g$ 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 Figure 6, where each point of $\text{G}(\mathbb{Z}^n)$ is labelled with its distance to the origin (see example Figure 14.a).

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 Figure 5. We scan each point p_1 of $\text{G}(B^{-1}(O, r))$, which by translation of vector $\overrightarrow{v}^g$ gives p_2. Values $d^2_E(O, p_1)$ and $d^2_E(O, p_2)$ are read in CT^g. We have

$$R = \max \left\{ d^2_E(O, p_2) \ : \ p_2 = p_1 + \overrightarrow{v}^g \ , \ p_1 \in \text{G}(B^{-1}(O, r)) \right\} \ , \text{ so} \tag{6}$$

$$R = \max \left\{ d^2_E(O, p_1 + \overrightarrow{v}^g) \ : \ p_1 \in \text{G}(B^{-1}(O, r)) \right\} \ . \tag{7}$$

This process can be efficiently implemented (see Figure 7), because all the covering relations (r, R) in a direction $\overrightarrow{v}^g$ can be detected during the same scan (lines 2–7). To remain in the bounds of the CT^g image, the x scan is limited to $L - \overrightarrow{v}^g_x - 1$ (where $\overrightarrow{v}^g_x$ is the x component of $\overrightarrow{v}^g$). For each point p_1, we look for the corresponding radius r_1 which is $CT^g[p_1] + 1$ by (4). Then we look for the radius r_2 of the ball passing via the point p_2. Its value is $CT^g[p_2] + 1 = CT^g[p_1 + \overrightarrow{v}^g] + 1$, by (4). During the scan, we keep in $Lut[\overrightarrow{v}^g][r_1]$ the greatest value found for r_2, which at the end, is R by (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 $Lut[\overrightarrow{v}^g][r_a] >$

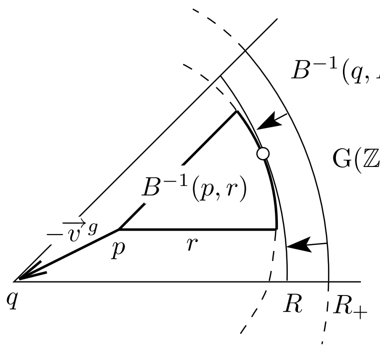

Fig. 4. Covering test on two balls restricted to $G(\mathbb{Z}^2)$.

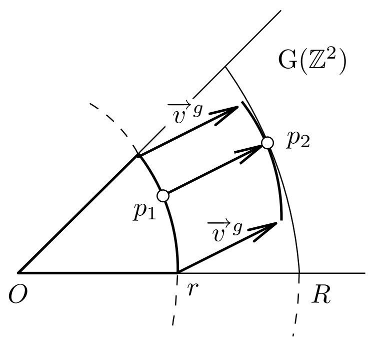

Fig. 5. Translated covering test on CT^g.

Procedure CompCTg (L, CT^g) ;
for $x_1 = 0$ **to** $L - 1$, **for** $x_2 = 0$ **to** x_1 , ... , **for** $x_n = 0$ **to** x_{n-1} **do**
    $CT^g[x_1, ..., x_n] = x_1^2 + \cdots + x_n^2$;

Fig. 6. Fast Cone Distance Transform. Input: L the side length. Output: CT^g the L^n distance image to the origin for d_E^2.

$Lut[\overrightarrow{v}^g][r_b]$; 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, $Lut[\overrightarrow{v}^g][r_b]$ should at least equal $Lut[\overrightarrow{v}^g][r_a]$, building this way a compatible total order (Figure 7, lines 8–10).

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

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

So as to increase R_{Known} to a given R_{Target}, we propose to test each ball $B(O, R)$, where $R > R_{Known}$, each time extracting its DT and then its MA, 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 $\mathcal{M}^g_{Lut}$ enables 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, then $\mathcal{M}^g_{Lut}$ is not sufficient to properly extract the MA, since by construction $B(O, R)$ covers $B^{-1}(p, DT^g[p])$. In this case we add a new vector $\overrightarrow{Op}$ in $\mathcal{M}^g_{Lut}$ (and keep R for further usage, see §4.2). This vector is necessary and sufficient to remove p from the MA of the ball $B(O, R)$ because the current $\mathcal{M}^g_{Lut}$ is validated until

Procedure CompLutCol (CT^g, L, $\overrightarrow{v}^g$, R_{max}, $Lut[\overrightarrow{v}^g]$) ;

for $r = 0$ **to** R_{max} **do** $Lut[\overrightarrow{v}^g][r] = 0$; *// Initialize* $Lut[\overrightarrow{v}^g]$ *to* 0.
for $x_1 = 0$ **to** $L - v^g_{x_1} - 1$, **for** $x_2 = 0$ **to** x_1 , $\ldots$, **for** $x_n = 0$ **to** x_{n-1} **do**
{
  $r_1 = CT^g[x_1, ..., x_n] + 1$; *// Radius of the ball where* p_1 *is located,*
  $r_2 = CT^g[(x_1, ..., x_n) + \overrightarrow{v}^g] + 1$; *// same for* p_2.
  if $r_1 \leq R_{max}$ **and** $r_2 > Lut[\overrightarrow{v}^g][r_1]$ **then** $Lut[\overrightarrow{v}^g][r_1] = r_2$;
}
$r_b = 0$;
for $r_a = 0$ **to** R_{max} **do**
  if $Lut[\overrightarrow{v}^g][r_a] > r_b$ **then** $r_b = Lut[\overrightarrow{v}^g][r_a]$ **else** $Lut[\overrightarrow{v}^g][r_a] = r_b$;

Fig. 7. *Lut* Column Computation. Input: CT^g the cone, L the side length, $\overrightarrow{v}^g$ the direction of the search, R_{max} the greatest radius value to be verified in *Lut*. Output: the column $Lut[\overrightarrow{v}^g]$ is filled with the correct values.

Procedure CompLutMask (L, $\mathcal{M}^g_{Lut}$, R_{Known}, R_{Target}, Lut) ;

CompCTg (L, CT^g) ;
for each $\overrightarrow{v}^g$ **in** $\mathcal{M}^g_{Lut}$ **do** CompLutCol (CT^g, L, $\overrightarrow{v}^g$, R_{Target}, $Lut[\overrightarrow{v}^g]$) ;
for $R = R_{Known} + 1$ **to** R_{Target} **do**
{
  for $x_1 = 0$ **to** $L - 1$, **for** $x_2 = 0$ **to** x_1 , $\ldots$, **for** $x_n = 0$ **to** x_{n-1} **do**
    if $CT^g[x_1, ..., x_n] \leq R$
      then $DT^g[x_1, ..., x_n] = 1$
      else $DT^g[x_1, ..., x_n] = 0$; *// Copy* $\mathrm{G}(B(R))$ *to* DT^g
  CompSEDTg (L, DT^g) ;
  for $x_1 = 1$ **to** $L - 1$, **for** $x_2 = 0$ **to** x_1 , $\ldots$, **for** $x_n = 0$ **to** x_{n-1} **do**
    if $DT^g[x_1, ..., x_n] \neq 0$ **and** IsMAg ($(x_1, ..., x_n), \mathcal{M}^g_{Lut}, Lut, DT^g$) **then**
    {
      $\mathcal{M}^g_{Lut} = \mathcal{M}^g_{Lut} \cup (x_1, ..., x_n\ ;\ R)$; *// Insert the new vector*
      CompLutCol (CT^g, L, $(x_1, ..., x_n)$, R_{Target}, $Lut[x_1, ..., x_n]$) ;
      if IsMAg ($(x_1, ..., x_n)$, $\mathcal{M}^g_{Lut}$, Lut, DT^g) **then error** ;
    }
}

Fig. 8. Full $\mathcal{M}^g_{Lut}$ and *Lut* Computation. Input: L the side length, $\mathcal{M}^g_{Lut}$, R_{Known} and R_{Target}. Output: *Lut*, $\mathcal{M}^g_{Lut}$ and R_{Target}. At first call, $\mathcal{M}^g_{Lut}$ and R_{Known} must be set to $\emptyset$ and 0 respectively. After exit, R_{Known} must be set to R_{Target}.

Function IsMAg (p, $\mathcal{M}^g_{Lut}$, Lut, DT^g) ;

for each $\overrightarrow{v}^g$ **in** $\mathcal{M}^g_{Lut}$ **do**
  if $p - \overrightarrow{v}^g \in \mathrm{G}(\mathbb{Z}^n)$ **then** *// Test only in* $\mathrm{G}(\mathbb{Z}^n)$.
    if $DT^g[p - \overrightarrow{v}^g] \geq Lut[\overrightarrow{v}^g][DT^g[p]]$ **then return false** ;
return true ;

Fig. 9. Fast extraction of MA points from $\mathrm{G}(B)$. Input: p the point to test, $\mathcal{M}^g_{Lut}$ 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.

$R-1$; thus it enables to find all the direct balls covering $B^{-1}(p, DT^g[p])$ of radii lower or equal to $R-1$. So, the only direct ball which is not tested is the only ball of radius R: $B(O, R)$ itself. This ball is in direction $\overrightarrow{pO}$ from p and must be searched by $\mathcal{M}^g_{Lut}$ to remove p. Since $\mathcal{M}_{Lut}$ is G-symmetric, $B(O, R)$ is detected by adding $\overrightarrow{Op}$ in its generator.

After having added the vector, we compute the corresponding new column in *Lut*. Then, we ensure that this new $\mathcal{M}_{Lut}$ is sufficient to remove p. This is actually a consistency test of the *Lut* column computation algorithm of Figure 7, because we are sure that the new $\mathcal{M}_{Lut}$ is correct.

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 $\mathcal{M}^g_{Lut}$ is finished when R reaches R_{Target}.

The full algorithm, presented in Figure 8, uses an adapted version of MA extraction (see Figure 9), working on $G(\mathbb{Z}^n)$ with $\mathcal{M}^g_{Lut}$ in a single scan. Note also that the computation of DT^g (function CompSEDTg called Figure 8, 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 $G(B(O, R))$ set, but not the correct DT^g labels (see Figure 14, where values of (a) differ from (b)).

4 Results for SEDT

4.1 Complexity

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 increase 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 [16], which provides exact results and can be relatively easily adapted to $G(\mathbb{Z}^n)$. In particular, backward scans can be suppressed [13, §6.5.2]. Note that SEDT on a ball is the worst case for the complexity of [16], and that optimised algorithms [8, 9] are noticeably more efficient for large radii.

The complexity in $\mathbb{Z}^n$ of CompSEDTg for a ball of radius R is $O(n.R^n)$ with [8, 9] or $O(n.R^{n+1})$ with [16]. The complexity of CompLutCol is $O(2.R^n)$ (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 $O(k.R^n)$ in the worst case, that is to say, when p is detected as an 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 Figure 8) is thus the complexity of 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 Figures 10 and 13. Computing the $\mathcal{M}^g_{Lut}$ shown Figure 10 takes 590s, while computing one corresponding *Lut* column takes 0.004s, for $L = 400$ and from $R_{Known} = 0$ to

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

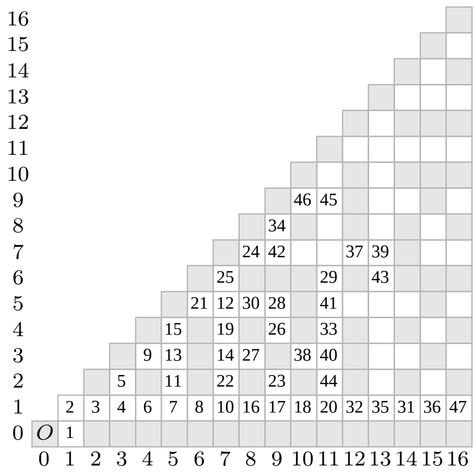

Fig. 11. Representation of $\mathcal{M}^g_{Lut}$ points (values show appearance ranks, white squares are visible points, grey squares non-visible).

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

Fig. 12. Beginning of Lut for $\mathbb{Z}^2$ (radius r, next columns $Lut[\overrightarrow{v}^g][r]$).

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

$R_{Target} = 128\,200$ (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) $\mathcal{M}^g_{Lut}$ should thus be saved for further re-usage.

The memory required to store *Lut* is $m.R.e$, where m is the number of columns in $\mathcal{M}^g_{Lut}$ for R, and e is the size of one long integer (to store d_E^2 values). In Figures 10 and 13 we can see that m grows slowly with R. 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 Figure 10 is 23 MB.

Memory can be saved by storing only possible values of d_E^2. The set of possible values in 2D is $S = \{\, a^2 + b^2 \leqslant R \,:\, a, b \in [0\,..\,R] \,\}$. The *Lut* entries are then accessed by $Lut[\overrightarrow{v}^g][index[r]]$, 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. The gain for *Lut* corresponding to Figure 10 is about 78% with only 5.1 MB to store. The same holds in 3D, but in lesser proportion. On the contrary in 4D and higher dimensions, any positive integer can be decomposed in sum of four (or more) squares (Lagrange thm., see [7, §20.5]), so that no space can be saved in this manner.

4.2 Extracting medial axis

A sample usage of the *Lut* given Figure 12 and formula (5) is : a point valued 4 on *DT* is not an MA point if, following third entry in table, it has at least a $(1,0)$-neighbour $\geqslant 6$, or a $(1,1)$-neighbour $\geqslant 9$, or a $(2,1)$-neighbour $\geqslant 14$, etc. The table is compressed by showing only possible radii r.

In Figures 10 and 13 are given the vectors of $\mathcal{M}^g_{Lut}$ in 2D and 3D respectively, and also their appearance radius R during CompLutMask. Keeping this radius is important because it allows 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 $\mathcal{M}^{R_{max}}_{Lut} = \{\, (\overrightarrow{v}; R) \in \mathcal{M}_{Lut} \,:\, R < R_{max} \,\}$ as the test neighbourhood to detect all MA points. In fact, CompLutMask garanties that $\mathcal{M}^{R_{max}}_{Lut}$ is necessary and sufficient up to $R_{Known} = R_{max} - 1$ in CT^g (as a radius of direct ball), thus by (4), up to R_{max} in *DT* (as a radius of reverse ball). For example in Figure 10, if $R_{max} = 101$ on *DT*, then the test neighbourhood will be limited to $(1,0)$-neighbours and $(1,1)$-neighbours.

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 $\mathcal{M}^g_{Lut}$, computed once for all, is already stored. The subset $\mathcal{M}^{R_{max}}_{Lut}$ is then used to extract MA, which is initialized to shape points. To minimize memory usage, we propose to allocate only one *Lut* column, instead of computing for R_{max} and $\#\mathcal{M}^{R_{max}}_{Lut}$ the whole *Lut*, which might be very large as seen in §4.1 : for each vector $\overrightarrow{v}^g$ in $\mathcal{M}^{R_{max}}_{Lut}$, we overwrite the previous column using CompLutCol, then reject from MA all the points which do not fulfill (5) with the G-symmetries of $\overrightarrow{v}^g$. This way, the MA set often decrease extremely fast at each step, thus accelerating the computation.

4.3 Properties

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 (even if the labels of the pixels on the DT are generally different). 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 > 1 exist in 2D and 3D because the sum of two or three squares does not fill $\mathbb{N}$. All the balls are different for dimension $n \geqslant 4$ because of Lagrange theorem; we think that this might have implications over properties of $\mathcal{M}_{Lut}$ and Lut which are linked to equivalence classes.

Our algorithm CompLutCol in Figure 7 gives the low bound of each equivalence class. We remark that the values published in [5] correspond to the high bounds; in that sense, the two tables must be considered as equivalent. Figure 10 also confirms the 3×3 test neighbourhood used in [5] for radii less than 80 in 2D, because the third direction only appears for $R = 101$.

We illustrate in Figure 14 the appearance of the direction $(2, 1)$ in $\mathcal{M}_{Lut}$ for $R = 101$ in $\mathbb{Z}^2$. The radius $R = 101$ of a direct ball (Figure 14.a) corresponds by (4) to radius $R' = 101 + 1$ of reverse ball. Since equivalence class of 102 is $[102, 103, \underline{104}]$, CompSEDTg labels O to 104 (Figure 14.b). When extracting MA with 2 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 (Figure 14.c,d), while it is overlapped in direction (2,1) (Figure 14.e).

Our experiments in 2D and 3D show that $\mathcal{M}_{Lut}$ is not bounded for d_E^2, unlike chamfer distances (see [14]). Figure 11 geometrically represents the set of vectors in $\mathcal{M}_{Lut}^g$ from Figure 10 with their rank of appearance. While layout seems random, one can note that all $\mathcal{M}_{Lut}$ 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 [18]). When carrying on computation of $\mathcal{M}_{Lut}^g$ with CompLutMask, all visible points seems to be gradually detected, while non-visible points never are. We therefore propose the conjecture:

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

These properties for d_E^2 are very different from those of chamfer distances (see [14]), where $\mathcal{M}_{Lut}$ are always bounded, Lut are bounded in most cases, and non-visible points may appear in $\mathcal{M}_{Lut}$. We think this is linked to the number of normals of the balls, which is unbounded for infinite Euclidean balls, while bounded for chamfer balls.

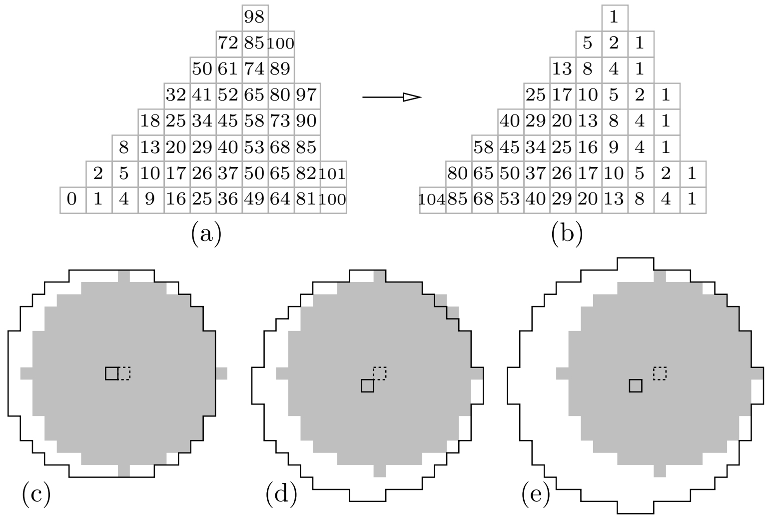

Fig. 14. Appearance of vector $(2, 1)$ in $\mathcal{M}_{Lut}$ for $R = 101$ in $\mathbb{Z}^2$. In (a), $B(101)$ is obtained using points $\leqslant 101$ from CT^g, and gives after SEDT, $B^{-1}(104)$ in (b), on which MA is extracted. In (c), $B^{-1}(65)$ (in gray) is not overlapped by $B^{-1}(80)$ in 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 $(2, 1)$.

5 Conclusion

The computation of the medial axis (MA) from the squared Euclidean distance transform (SEDT) is detailed for arbitrary dimension. The principle of MA extraction using look-up tables (*Lut*) was already published for d_E^2 in 2D for small values and 3×3 neighbourhood in [5], but no general method to compute them was given. We have introduced the mask $\mathcal{M}_{Lut}$, which stores the test neighbourhood used during the MA extraction. We showed that, in the general case, the mask $\mathcal{M}_{Lut}$ is greater than just the 3^n neighbourhood. We have presented and justified efficient algorithms which compute both *Lut* and $\mathcal{M}_{Lut}$ for d_E^2. Our algorithms certify that $\mathcal{M}_{Lut}$ is sufficient up to a given ball radius. We give a sample *Lut* table in 2D for comparison with [5]. We give two sets of $\mathcal{M}_{Lut}^g$ 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 show that, in the case of d_E^2, the neighbourhood $\mathcal{M}_{Lut}$ to test is a set of visible points. Unlike seen in the case of chamfer distances in [14], 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.

References

1. C. Arcelli and G. Sanniti di Baja. Finding local maxima in a pseudo-Euclidean distance transform. *Comp. Vision, Graphics and Image Proc.*, 43:361–367, 1988.
2. H. Blum. A transformation for extracting new descriptors of shape. In W. Wathendunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380, Cambridge, 1967. MIT Press.
3. G. Borgefors. Centres of maximal disks in the 5-7-11 distance transform. In 8^{th} *Scand. Conf. on Image Analysis*, pages 105–111, Tromsø, Norway, 1993.
4. G. Borgefors and I. Nyström. Efficient shape representation by minimizing the set of centres of maximal discs/spheres. *Pat. Rec. Letters*, 18:465–472, 1997.
5. G. Borgefors, I. Ragnemalm, and G. Sanniti di Baja. The Euclidean Distance Transform : finding the local maxima and reconstructing the shape. In 7^{th} *Scand. Conf. on Image Analysis*, volume 2, pages 974–981, Aalborg, Denmark, 1991.
6. P.E. Danielsson. Euclidean distance mapping. *Comp. Graphics and Image Proc.*, 14:227–248, 1980.
7. G.H. Hardy and E.M. Wright. *An introduction to the theory of numbers*. Oxford University Press, fifth edition, October 1978.
8. T. Hirata. A unified linear-time algorithm for computing distance maps. *Information Proc. Letters*, 58:129–133, 1996.
9. A. Meijster, J.B.T.M. Roerdink, and W.H. Hesselink. A general algo. for comp. distance trans. in linear time. In Goutsias and Bloomberg, editors, *Math. Morph. and its App. to Image and Signal Proc.*, pages 331–340. Kluwer, 2000.
10. F. Nilsson and P.E. Danielsson. Finding the minimal set of maximum disks for binary objects. *Graph. Models and Image Proc.*, 59(1):55–60, 1997.
11. J.L. Pfaltz and A. Rosenfeld. Computer representation of planar regions by their skeletons. *Comm. of ACM*, 10:119–125, feb 1967.
12. I. Ragnemalm. The Euclidean distance transform in arbitrary dimensions. *Pat. Rec. Letters*, 14(11):883–888, 1993.
13. E. Remy. *Normes de chanfrein et axe médian dans le volume discret*. PhD, Univ. de la Méditerranée, Aix-Marseille 2, Dec 2001.
14. E. Remy and E. Thiel. Medial axis for chamfer distances: computing look-up tables and neighbourhoods in 2D or 3D. *Pat. Rec. Letters*, 23(6):649–661, April 2002.
15. A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of ACM*, 13(4):471–494, 1966.
16. T. Saito and J.I. Toriwaki. New algorithms for Euclidean distance trans. of an n-dim. digitized picture with applications. *Pat. Rec.*, 27(11):1551–1565, 1994.
17. G. Sanniti di Baja and E. Thiel. A skeletonization algorithm running on path-based distance maps. *Image and Vision Computing*, 14(1):47–57, Feb 1996.
18. 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` .