ADAPTABLE MATHEMATICAL MORPHOLOGY IN D DIMENSIONS USING THE SEPARABLE EUCLIDEAN DT IN $D+1$ DIMENSIONS

Olivier Cuisenaire

Signal Processing Institute (ITS), Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
email: olivier.cuisenaire@epfl.ch, web: itswww.epfl.ch/∼cuisenai

ABSTRACT

Locally adaptable mathematical morphology (AMM) uses circular structuring elements whose sizes can vary arbitrarily over the image plane. In this paper, we present an efficient algorithm to implement the dilation, erosion, closing and opening operators in arbitrary dimensions. The core of the method relies on adapting the separable Euclidean distance transformation (DT) introduced by Maurer in [IEEE Trans. PAMI, 25(2):265-270,2003]. The algorithm is both more accurate and significantly faster than the previously published method.

1. INTRODUCTION

In [1, 2], we developed adaptable mathematical morphology, which extends the traditional translation-invariant binary morphology so that the size of the structuring element can be specified for each pixel of the image, provided that they are shaped like balls of a distance metric. The implementation uses a special raster-scanning similar to Danielsson's Euclidean DT algorithm [3] in 2 dimensions and Ragnemalm's corner DT [4] for higher dimensions.

When using circular or spherical balls, i.e. when one considers the Euclidean metric, this approach is only an approximate solution and can mislabel some pixels. Also, for images in more than 2 dimensions, it requires 2^D scans over the image which leads to a high computational cost. In this paper, we show that these AMM operators can be computed both more precisely and faster using a variation of the separable Euclidean DT algorithms proposed by Maurer [5], Meijster [6] and Hirata [7].

The paper is organized as follows. In section 2, we recall the basic principles and algorithm of AMM from [1, 2]. Section 3 discusses the limits of this raster scanning algorithm. Section 4 presents the new algorithms for AMM dilation and closing which overcome those limits. Finally, section 5 discusses the computational complexity of the method and compares it experimentally with the previous approach.

2. AMM PRINCIPLES

In [1, 2], we laid out the principles of AMM. The adaptable dilation of an object X in a binary image I in D dimensions by structuring elements whose sizes are stored in an image S is defined as

$$X \oplus S = \{\mathbf{x}+\mathbf{h} : \mathbf{x} \in X, \|\mathbf{h}\| < S(\mathbf{x})\} \tag{1}$$

where pixel locations are represented by D-dimensional vectors $\mathbf{v} = (v_1, v_2, ..., v_D)$ and we consider the Euclidean metric

$$\|\mathbf{v}\| = (\sum_{d=1}^{D} v_d^2)^{\frac{1}{2}} \tag{2}$$

for circular and spherical structuring elements. The adaptable erosion is defined by duality as

$$X \ominus S = (X^c \oplus S)^c \tag{3}$$

where $X^c = \{\mathbf{x} : \mathbf{x} \notin X\}$ is the set complement of X. For the opening and closing operators, we define a reflected dilation operator $\check{\oplus}$ as

$$X \check{\oplus} S = \{\mathbf{y} : D_X(\mathbf{y}) < S(\mathbf{y})\} \tag{4}$$

where D_X is the Euclidean distance transformation of X, i.e.

$$D_X(\mathbf{p}) = \min_{\mathbf{x} \in X}(\|\mathbf{p}-\mathbf{x}\|) \tag{5}$$

The AMM closing X^S and opening X_S are then defined as

$$X^S = (X \check{\oplus} S) \ominus S \tag{6}$$
$$X_S = (X \check{\ominus} S) \oplus S \tag{7}$$

In [1, 2], we prove that these operators are indeed morphological closing and opening filters. X^S is increasing, anti-extensive and idempotent, X_S is increasing, extensive and idempotent.

From the implementation point of view, the most complex operator is the adaptive dilation of equation (1). In [2], it is computed by noticing that

$$\begin{aligned} X \oplus S &= \{\mathbf{x}+\mathbf{h} : \mathbf{x} \in X, \|\mathbf{h}\| < S(\mathbf{x})\} \\ &= \{\mathbf{x}+\mathbf{h} : \mathbf{x} \in X, \|\mathbf{h}\| - S(\mathbf{x}) < 0\} \\ &= \{\mathbf{y} : \exists\, \mathbf{x} \in X, \|\mathbf{y}-\mathbf{x}\| - S(\mathbf{x}) < 0\} \\ &= \{\mathbf{y} : D_{X,S}(\mathbf{y}) < 0\} \end{aligned} \tag{8}$$

where $D_{X,S}$ is a modified distance transformation defined as

$$D_{X,S}(\mathbf{p}) = \min_{\mathbf{x} \in X}(\|\mathbf{p}-\mathbf{x}\| - S(\mathbf{x})) \tag{9}$$

This modified DT is computed by generating

$$\mathbf{V}(\mathbf{p}) = \arg\min_{\mathbf{x} \in X}(\|\mathbf{p}-\mathbf{x}\| - S(\mathbf{x})) \tag{10}$$

which is the Voronoi partition of the image plane corresponding to the distance $D_{X,S}$. The algorithm initializes with $\mathbf{V}(\mathbf{p}) = \mathbf{p}$ and $D_{X,S}(\mathbf{p}) = -S(\mathbf{p})$ for object pixels and $\mathbf{V}(\mathbf{p}) = (\infty, \infty)$ and $D_{X,S}(\mathbf{p}) = \infty$ for the background. The distances are then propagated from neighbor to neighbor. $\mathbf{V}(\mathbf{p}+\mathbf{n})$ replaces $\mathbf{V}(\mathbf{p})$ as the object pixel closest to $\mathbf{p}$ if

$$\|\mathbf{p} - \mathbf{V}(\mathbf{p}+\mathbf{n})\| - S(\mathbf{V}(\mathbf{p}+\mathbf{n})) < D_{X,S}(\mathbf{p}) \tag{11}$$

In 2D, this is implemented using a raster scanning similar to Danielsson's Euclidean DT algorithm [3]. Firstly, The image is

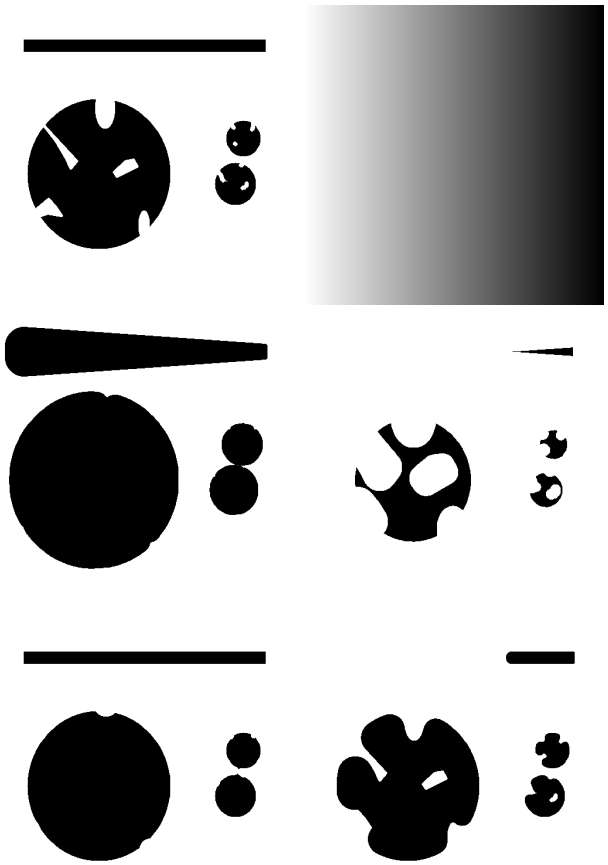

Figure 1: AMM operators: **(top-left)** Original image I of size 512×512 with object X in black. **(top-right)** Structuring element image S chosen as $S(\mathbf{p}) = (512 - p_x)/14$. **(center-left)** Dilation $X \oplus S$. **(center-right)** Erosion $X \ominus S$ **(bottom-left)** Closing X^S **(bottom-right)** Opening X_S.

scanned line by line, from top to bottom. Each line is first scanned from left to right considering (11) for the left and up neighbors, then from right to left considering the right neighbor. Secondly, the dual scan from bottom to top and right to left is performed.

In $D \geq 3$ dimensions, scans similar to Ragnemalm's corner Euclidean DT [4] are used. It uses 2^D scans from each of the corners to its opposite, considering the D direct neighbors on the side of the originating corner at each scan.

The effect of the AMM operators on a synthetic image is illustrated at figure 1. There are numerous possible applications. Essentially, when the projective geometry in the image acquisition process makes translation invariance an incorrect assumption, AMM becomes a relevant tool. This includes traffic control cameras where vehicles higher in the image are further away and appear smaller than those at the bottom, weather satellite imagery where both the wide camera angle and the earth curvature introduce geometric distortion, ... AMM also offers a practical implementation of several of Roerdink's group morphologies [8]. It is also possible to use AMM for adaptive morphology where the SE sizes depend on the local pixel values. This is particularly relevant for range imagery. Finally, in medical imaging anatomical knowledge can drive the choice of SE sizes that are locally adapted to the organs imaged. The ability to work in more than 2 dimensions is critical in medical imaging where 3D and 4D image modalities are common.

3. LIMITATIONS OF THE RASTER SCANNING ALGORITHM

While it is intuitive and relatively easy to implement, the method described in section 2 for computing $D_{X,S}$ and $X \oplus S$ suffers from a number of limitations.

Firstly, storing the whole vectorial image $\mathbf{V}$ is quite memory consuming, as it requires storing D coordinates per pixels in D dimensions. Typically, this requires more memory than the image itself.

Secondly, the raster scanning similar to Ragnemalm's corner DT [4] requires 2^D scans over the image, which becomes time-consuming for $D \geq 3$ dimensions.

Thirdly, equation (11) includes a square root computation which cannot be avoided. While all fast Euclidean DT algorithms use the square of the Euclidean distance during propagation and only compute one square root per pixel as post-processing, this algorithm uses the square root operation at every propagation test. That is $D.2^D$ times for the corner DT in D dimensions.

Finally, the result is only an approximation of $D_{X,S}$. In order to be exact, the tiles of the Voronoi partition $\mathbf{V}$, i.e. the the sets

$$T(\mathbf{x}) = \{\mathbf{p} : \mathbf{V}(\mathbf{p}) = \mathbf{x}\} \tag{12}$$

should be connected sets. While in [2] we prove this property for the continuous plane, it is not true on a discrete lattice. For instance, using direct neighbors only, if the object is $X = \{(0,3),(3,0),(2,2)\}$ and $S(\mathbf{p})$ is a constant s for all pixels, the distance $D_{X,S}((0,0))$ is mistakenly computed to be $3 - s$ instead of its true value $\sqrt{8} - s$. Indeed, the information from object pixel $(2,2)$ can not reach $(0,0)$ because its direct neighbors $(1,0)$ and $(0,1)$ are closer to $(3,0)$ and $(0,3)$ than $(2,2)$, respectively.

Whatever the neighborhood used, there can be a few pixels where $D_{X,S}$ is slightly overestimated. Therefore, a few pixels of $X \oplus S$ can be mislabelled as belonging to $(X \oplus S)^c$.

4. FAST AMM ALGORITHMS

In this paper, we overcome these limitations by proposing a new algorithm to compute the adaptive dilation. We rewrite (1) as

$$\begin{aligned} X \oplus S &= \{\mathbf{y} : \exists \mathbf{x} \in X, \|\mathbf{y} - \mathbf{x}\| < S(\mathbf{x})\} \\ &= \{\mathbf{y} : \exists \mathbf{x} \in X, \|\mathbf{y} - \mathbf{x}\|^2 - S(\mathbf{x})^2 < 0\} \\ &= \{\mathbf{y} : \exists \mathbf{x} \in X, \|\mathbf{y} - \mathbf{x}\|^2 + (M^2 - S(\mathbf{x})^2) < M^2\} \end{aligned} \tag{13}$$

where M can be any number that respects

$$M \geq \max_{\mathbf{p} \in I}(S(\mathbf{p})) \tag{14}$$

With such a choice, $(M^2 - S(\mathbf{p})^2)$ is positive for all $\mathbf{p}$. Then, we can interpret the expression $\|\mathbf{y} - \mathbf{x}\|^2 + (M^2 - S(\mathbf{x})^2)$ of (13) as the square of a Euclidean distance. Instead of considering the pixel locations $\mathbf{x}$ and $\mathbf{y}$ as D-dimensional vectors, we see them as $(D+1)$-dimensional vectors with their $(D+1)$th component equal to 0. Then, we apply a transform $f_S(.)$ to the set X which moves its points $\mathbf{x}$ "out of plane".

$$f_S(X) = \{\mathbf{x} + \mathbf{e}_\perp . \sqrt{M^2 - S(\mathbf{x})^2} : \mathbf{x} \in X\} \tag{15}$$

where $\mathbf{e}_\perp$ is the unit vector in the $(D+1)$th dimension. After such a transform, equation (13) can be rewritten as

$$\begin{aligned} X \oplus S &= \{\mathbf{y} : y_{D+1} = 0, \exists \mathbf{x}' \in X' = f_S(X), \|\mathbf{y} - \mathbf{x}'\|^2 < M^2\} \\ &= \{\mathbf{y} : y_{D+1} = 0, D_{X'}(\mathbf{y}) < M\} \end{aligned} \tag{16}$$

for all $\mathbf{p} \in I$ **do**
  if $\mathbf{p} \in X$ **then**
    $D^2_{X'}(\mathbf{p}) \leftarrow M^2 - S(\mathbf{p})^2$
  else
    $D^2_{X'}(\mathbf{p}) \leftarrow M^2$

ComputeEDT($D^2_{X'}$)

for all $\mathbf{p} \in I$ **do**
  if $D^2_{X'}(\mathbf{p}) < M^2$ **then**
    $\mathbf{p} \in X \oplus S$
  else
    $\mathbf{p} \in (X \oplus S)^c$

Algorithm 1: $X \oplus S$

for all $\mathbf{p} \in I$ **do**
  if $\mathbf{p} \in X$ **then**
    $D^2_X(\mathbf{p}) \leftarrow 0$
  else
    $D^2_X(\mathbf{p}) \leftarrow M^2$

ComputeEDT(D^2_X)

for all $\mathbf{p} \in I$ **do**
  if $D^2_X(\mathbf{p}) < S(\mathbf{p})^2$ **then**
    $\mathbf{p} \in X \check{\oplus} S$
  else
    $\mathbf{p} \in (X \check{\oplus} S)^c$

Algorithm 2: $X \check{\oplus} S$

for $d = 1 \rightarrow D$ **do**
  for $i_1 = 1 \rightarrow N_1$ **do**
    ...
    for $i_{d-1} = 1 \rightarrow N_{d-1}$ **do**
      for $i_{d+1} = 1 \rightarrow N_{d+1}$ **do**
        ...
        for $i_D = 1 \rightarrow N_D$ **do**
          Compute1DT($D_e, d, i_1, ..., i_{d-1}, i_{d+1}, ..., i_D$)
        ...
    ...

Procedure 1: ComputeEDT(D_e)

$k \leftarrow 0$
for $j = 1 \rightarrow N_d$ **do**
  $\mathbf{p_j} \leftarrow (i_1, ..., i_{d-1}, j, i_{d+1}, ..., i_D)$
  if $k < 2$ **then**
    $k \leftarrow k+1$, $g_k \leftarrow D_e(\mathbf{p_j})$, $h_k \leftarrow j$
  else
    while $k \geq 2$ **and** $(j - h_{k-1}).g_k - (j - h_k).g_{k-1} - (h_k - h_{k-1}).D_e(\mathbf{p_j}) - (j - h_{k-1}).(j - h_k).(h_k - h_{k-1}) > 0$ **do**
      $k \leftarrow k-1$
    $k \leftarrow k+1$, $g_k \leftarrow D_e(\mathbf{p_j})$, $h_k \leftarrow j$

$m \leftarrow k$, $k \leftarrow 1$
for $j = 1 \rightarrow N_d$ **do**
  while $k < m$ **and** $g_k + (h_k - j)^2 > g_{k+1} + (h_{k+1} - j)^2$ **do**
    $k \leftarrow k+1$
  $D_e(\mathbf{p_j}) \leftarrow g_k + (h_k - j)^2$

Procedure 2: Compute1DT($D_e, d, i_1, ..., i_{d-1}, i_{d+1}, ..., i_D$)

where $D_{X'}$ is the $(D+1)$-dimensional Euclidean DT applied on the set $X' = f_S(X)$. Intuitively, in 2D, this means that instead of using circular structuring elements of varying radiuses $S(\mathbf{x})$, we use 3D spherical structuring elements of constant radius M, but the object pixels have been moved out of the image plane at a distance $\sqrt{M^2 - S(\mathbf{x})^2}$.

The interest of this approach is that efficient implementations of the Euclidean DT have been the subject of a significant amount of research since Danielsson's paper in 1980. An in-depth review can be found in chapter 2 of [9].

On the other hand, if we want to work in the D-dimensional image and not in $(D+1)$ dimensions, we can not use many of the existing Euclidean DT algorithms. The propagation algorithm such as the raster scanning or ordered propagation methods explored in [2] do not work. Indeed, while in $D+1$ dimensions, the tiles

$$T(\mathbf{x}') = \{\mathbf{p} : \mathbf{x}' = \arg\min_{\mathbf{y}' in X'}(\|\mathbf{p} - \mathbf{y}'\|)\} \tag{17}$$

are convex (hyper)-polyhedra surrounding their translated object pixel $\mathbf{x}'$, in the D-dimensional image, the tiles

$$T(\mathbf{x}) = \{\mathbf{p} : \mathbf{x} = \arg\min_{\mathbf{y} in X}(\|\mathbf{p} - \mathbf{y}\|^2 + M^2 - S(\mathbf{y})^2)\} \tag{18}$$

are not. They are not even star-shaped and actually, $T(\mathbf{x})$ can be non empty and still not include $\mathbf{x}$.

Fortunately, this problem has been solved previously by Saito [10], Hirata [7], Meijster [6] and Maurer [5] who proposed separable Euclidean DT algorithms. The D-dimensional problem is split into D 1-dimensional problems. When processing the dth dimension, one produces an intermediate result

$$D_{X,d}(\mathbf{p}) = \min_{\mathbf{x} \in X_d(\mathbf{p})}(\|\mathbf{p} - \mathbf{x}\|) \tag{19}$$

$$X_d(\mathbf{p}) = \{\mathbf{x} = (x_1, ..., x_D) : \forall i > d, x_i = p_i, \} \cap X \tag{20}$$

where $X_d(\mathbf{p})$ is the set of object points with the same coordinates as $\mathbf{p}$ for $i > d$. Initially, $D_{X,0}$ is 0 for objects pixels and ∞ for background pixels. Then, the algorithms proceed iteratively by computing, for all $\mathbf{p}$,

$$D_{X,d}(\mathbf{p}) = \min_{\mathbf{q} \in L_d(\mathbf{p})}(\sqrt{D_{X,d-1}(\mathbf{q})^2 + \|\mathbf{p} - \mathbf{q}\|^2}) \tag{21}$$

$$L_d(\mathbf{p}) = \{\mathbf{q} : \forall i \neq d, p_i = q_i\} \tag{22}$$

where $L_d(\mathbf{p})$ is the line of pixels with all coordinates identical to $\mathbf{p}$'s except for the dth. After iterating over $d = 1 \rightarrow D$, we get $D_X = D_{X,D}$.

How (21) is implemented practically varies between authors. Obviously all of them get rid of the square root operation by computing $D^2_{X,d}$ instead of $D_{X,d}$. Also, they use the alignment of $\mathbf{p}$ and $\mathbf{q}$ to simplify $\|\mathbf{p} - \mathbf{q}\|$. Equation (21) becomes

$$D_{X,d}(\mathbf{p})^2 = \min_{\mathbf{q} \in L_d(\mathbf{p})}(D_{X,d-1}(\mathbf{q})^2 + (p_d - q_d)^2) \tag{23}$$

Beyond that, computations can be further accelerated by considering information from the other pixels in the same line to reduce the search space L_d over which the minimum is computed. Saito [10] proposes a heuristic method which speeds up the process practically but has the same asymptotical complexity as (21), i.e. in order to compute $D_{X,d}$ at the N_d points of a line, it requires $\circ(N_d^2)$ distance computations. The others [7, 5, 6] give very similar solutions that are optimal in the sense that N_d points are computed with a $\circ(N_d)$ complexity.

Explaining the details of these methods is beyond the scope of this paper. In Procedure 2, Compute1DT implements the algorithm described by Maurer [5]. Using this tool, it is simple to compute $X \oplus S$ and $X \check{\oplus} S$.

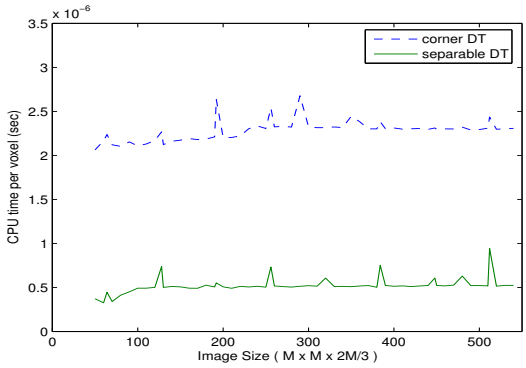

Figure 2: CPU times to compute the AMM closing of a 3D image of varying size using either the old method (corner DT) or the new algorithm (separable DT).

Algorithm 1 computes $X \oplus S$ for an image of size $N_1 \times N_2 \times \ldots \times N_D$. It uses (16) and the $(D+1)$-dimensional Euclidean DT. First, it considers the $(D+1)$th dimension for which the square of the distance is $M^2 - S(\mathbf{p})^2$ for object pixels and ∞ for background pixels. Then, it calls the ComputeEDT procedure to iterate over the remaining D dimensions of the image. Finally, it thresholds the result by M^2.

Algorithm 2 computes $X \check{\oplus} S$ using (4) and the D-dimensional Euclidean DT. First, it assigns a distance 0 to object pixels and ∞ to the background, then it calls the ComputeEDT procedure to iterate over the D dimensions of the image. Finally, it thresholds locally by $S(\mathbf{p})$.

In the initialization step of both algorithms, we use M^2 instead of ∞ without affecting the final result, since M^2 is larger than the threshold used at the end of the algorithm.

Finally, all AMM operators can be computed from $X \oplus S$ and $X \check{\oplus} S$ using equations (3), (6) and (7).

5. COMPUTATIONAL COMPLEXITY

The method described here has numerous advantages compared to the raster scanning of section 2 and its limitations discussed at section 3. Firstly, it does not need to compute explicitly the Voronoi diagram $\mathbf{V}$ which reduces the memory requirements to a minimum. Secondly, in addition to the initialization and threshold steps, it uses D scans over the image instead of 2^D, which is significantly lower for $D \geq 3$. Thirdly, it uses no square root operation whatsoever instead of $D.2^D$ per pixel. Finally, it computes the exact distance transformation and therefore the exact dilations $X \oplus S$ and $X \check{\oplus} S$.

Beyond these theoretical considerations, the computational cost was assessed experimentally on a Pentium IV computer at 3GHz, with 512 kB of cache memory. Varying the object X and structuring elements sizes S has no effect on the CPU times needed. In 2 dimensions, the new algorithm does not improve significantly on the old method beyond its improved accuracy. Typically, a 1000×1000 image is closed in 0.2 second by both algorithms.

Obviously, the competitive advantage of the new algorithm appears in $D \geq 3$ dimensions. At figure 2, the relationship between image size and execution time is assessed using a $3D$ test image of size $M \times M \times \frac{2}{3}M$ made of anisotropic voxels of size $1 \times 1 \times 1.5$, with M varying between 50 and 540. Both the corner DT approach and the new separable algorithm are used to implement the AMM closing. This experiment shows that the algorithm described in this paper is more than 4 times faster than the former approach. In both cases, the time required per voxel is approximately constant.

An interesting phenomenon appears for the separable DT. CPU times show spikes for $M = k.2^m$ with a large m. For these values, the algorithm is impaired by the inefficiency of the CPU's cache during the inter-slice scan. For instance, for a slice of $512 \times 512 = 2^{18}$ voxels, if voxels are coded with 4 bytes each, two pixels at the same location on consecutive slices have memory addresses sharing the same last 22 binary digits. Since those are used to compute the cache address, this leads to multiple cache conflicts during the inter-slice scan.

This problem can be solved in many ways, for instance by zero-padding the image so that the slices have one more line and column. Computing the closing for an image of size $512 \times 512 \times 342$ requires 222 or 86 seconds using the old or the new algorithm, respectively. This goes down to 218 and 48 seconds respectively for a $513 \times 513 \times 342$ zero-padded image.

6. CONCLUSION

We have proposed a new algorithm to implement adaptive mathematical morphology using (hyper)-spherical structuring elements in D dimensions. Contrarily to the approximate raster scanning algorithm used previously, it provides an exact result. In $D \geq 3$ dimensions, it is significantly faster than the previous method.

REFERENCES

[1] O. Cuisenaire, "Locally adaptable mathematical morphology," submitted to ICIP'05.

[2] O. Cuisenaire, "Locally adaptable binary mathematical morphology using distance transformations," Technical Report ITS-2004.30, Signal Processing Institute, EPFL, CH-1015 Lausanne, Switzerland, December 2004.

[3] P.E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, pp. 227–248, 1980.

[4] I. Ragnemalm, "The euclidean distance transformation in arbitrary dimensions," *Pattern Recognition Letters*, vol. 14, pp. 883–888, 1993.

[5] C.R. Maurer Jr, R. Qi, and V. Raghavan, "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[6] 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., 2000, pp. 331–340.

[7] T. Hirata, "A unified linear-time algorithm for computing distance maps," *Information Processing Letters*, vol. 58, no. 3, pp. 129–133, 1996.

[8] J.B.T.M Roerdink, "Group morphology," *Pattern Recognition*, vol. 33, pp. 877–895, 2000.

[9] O. Cuisenaire, *Distance transformations: fast algorithms and applications to medical image processing*, Ph.D. thesis, Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, October 1999.

[10] T. Saito and J.I. Toriwaki, "New algorithms for euclidean distance transformations of an n-dimensional digitised picture with applications," *Pattern Recognition*, vol. 27, no. 11, pp. 1551–1565, 1994.