Pattern Recognition Letters 32 (2011) 1356–1364

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Separable algorithms for distance transformations on irregular grids

Antoine Vacavant [a,*], David Coeurjolly [b,c], Laure Tougne [b,d]

[a] Clermont Université, Université d'Auvergne, ISIT F-63000, France
[b] Université de Lyon, CNRS, France
[c] Université Lyon 1, LIRIS, UMR5205 F-69622, France
[d] Université Lyon 2, LIRIS, UMR5205 F-69676, France

ARTICLE INFO

Article history:
Available online 24 November 2010

Keywords:
Squared Euclidean distance transformation
Irregular grids
Quadtree
Voronoi diagrams

ABSTRACT

In this article, we propose to investigate two extensions of the E^2DT (squared Euclidean Distance Transformation) on irregular isothetic grids (or $\mathbb{I}$-grids), such as quadtree/octree or run-length encoded d-dimensional images. We enumerate the advantages and drawbacks of the $\mathbb{I}$-CDT, based on the cell centres, and the ones of the $\mathbb{I}$-BDT, which uses the cell borders. One of the main problem we mention is that no efficient algorithm has been designed to compute both transforms in arbitrary dimensions. To tackle this problem, we describe in this paper two algorithms, separable in dimension, to compute these distance transformations in the two-dimensional case, and we show that they can be easily extended to higher dimensions.

1. Introduction

The distance transformation (Klette and Rosenfeld, 2004; Rosenfeld and Pfaltz, 1968) (DT) of a binary image consists in labeling each point of a discrete object $\mathcal{E}$ (*i.e.* foreground) with its shortest distance to the complement of $\mathcal{E}$ (*i.e.* background). This process has been widely studied and developed on regular grids (see Fabbri et al., 2008 for a survey). Some specific extensions of the DT to non-regular grids also exist, such as elongated grids (Chehadeh et al., 1996; Fouard and Malandain, 2005; Sintorn and Borgefors, 2004), quadtrees/octrees (Samet, 1990; Vörös, 2001), Face-Centered Cubic (FCC)/Body-Centered Cubic (BCC) grids (Fouard et al., 2007), *etc.* But, to our knowledge, no generic DT process has been designed on every kinds of image representations in two or higher dimensions.

This article deals with generalizing the DT computation on *irregular isothetic grids* (or $\mathbb{I}$-grid for short). In two dimensions (2-D), an $\mathbb{I}$-grid is defined by a partition of $\mathbb{R}^2$ with rectangular cells whose edges are aligned along the two axes. The quadtree decomposition and the RLE (Run Length Encoding) are examples of classical techniques in imagery which induce an $\mathbb{I}$-grid. Here, we focus our interest on generalizing techniques that compute the E^2DT (squared Euclidean DT) of a d-dimensional (d-D) binary image (Coeurjolly and Montanvert, 2007; Maurer et al., 2003; Saito and Toriwaki, 1994). This transformation is a very common way to analyze the shape of graphical objects for various applications (see Paglieroni, 1992 and references of Maurer et al. (2003), for more details). Many of those techniques can be linked to the computation of a discrete Voronoi diagram of the background pixels (Coeurjolly and Montanvert, 2007; Hesselink et al., 2005). A further application of our work would be to compute E^2DT in octree-based images, which are a common way to model very large-scale volumes (Samet, 1990).

In (Vacavant et al., 2008; Vacavant et al., 2009a), we introduced two kinds of extensions of the E^2DT on $\mathbb{I}$-grids. The first one is based on the distance between cell centres, while the second one computes the distance between a cell centre and the nearest foreground/background frontier. In this article, we make a complete review of each model, and we list their differences. We also recall previous work and present several techniques to compute them. In this review, we show that no efficient algorithm has been designed to compute both distance transformations in arbitrary dimensions. The methods that we introduce in this article are separable, *i.e.* they perform operations independently along the two axes and can be naturally extended to handle d-D images. Moreover, they can compute both E^2DT extensions we have introduced in our previous work.

This paper is organized as follows: We first recall in Section 2 previous work about DT on the $\mathbb{I}$-grid model, the two extensions of the E^2DT on these grids, with their advantages and drawbacks. Then, we present the mathematical and algorithmic tools we need to develop separable DT processes on $\mathbb{I}$-grid (Section 3). In Section 4, we describe two separable DT algorithms and we finally compare them (in terms of complexity and execution times), in Section 5.

* Corresponding author. Tel.: +33 4 78 77 23 38.
E-mail address: antoine.vacavant@iut.u-clermont1.fr (A. Vacavant).

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.11.010

2. Previous work about DT on $\mathbb{I}$grids

In this section, we first recall the concept of irregular isothetic grids ($\mathbb{I}$-grids) in 2-D, with the following definition (Coeurjolly and Zerarga, 2006):

Definition 1 (*2-D $\mathbb{I}$-grid*). Let P be a closed rectangular subset of $\mathbb{R}^2$. A 2-D $\mathbb{I}$-grid G is a tiling of P with non overlapping rectangular cells whose edges are parallel to the X- and Y-axis. The position of each cell is characterized by its centre point (its position) $(x_R, y_R) \in \mathbb{R}^2$ and its length along X- and Y-axis $(l_R^x, l_R^y) \in \mathbb{R}_+^{*2}$.

We consider this definition of 2-D $\mathbb{I}$-grids for the rest of the paper, and we will shortly show that our contribution is easily extensible to the d-D case. In our framework, we consider *labeled* $\mathbb{I}$-grids, *i.e.* each cell of the grid has a *foreground* or *background* label (its value is respectively "0" or "1" for example). For an $\mathbb{I}$-grid G, we denote by G_F and G_B the sets of foreground and background cells.

Here, we first recall the two extensions of the E^2DT proposed on $\mathbb{I}$-grids, for each cell $R \in G_F$:

$$\mathbb{I}\text{-CDT}(R) = \min_{R'} \left\{ d_e^2(p, p') : R' \in G_B \right\}, \quad (1)$$

$$\mathbb{I}\text{-BDT}(R) = \min_{s} \left\{ d_e^2(p, s) : s \in \mathcal{S} \right\}. \quad (2)$$

where $\mathcal{S}$ is the set of segments contained in the background/foreground frontier of the grid. The first extension, the Centre-based DT (Eq. (1)) (Vacavant et al., 2008) is based on the cell centres of the grid, where $d_e^2(p,p')$ the squared Euclidean distance between the centres $p = (x_R, y_R)$ and $p' = (x_{R'}, y_{R'})$. The Border-based DT (Eq. (2)) (Vacavant et al., 2009a) employs cell borders, and computes the shortest distance between a foreground cell centre and the background/foreground boundary. We sum up in Table 1 the advantages and the drawbacks of each model.

As mentioned in Table 1, we can compute these extensions thanks to the construction of two kinds of Voronoi diagrams (VD). We recall that the VD of a set of points $\mathcal{P} = \{p_i\}$ is a tiling of the plane into *Voronoi cells* (or *VD cells*) $\{C_{p_i}\}$ (de Berg et al., 2000; Voronoi, 1908). The $\mathbb{I}$-CDT may be computed with such a structure, as we discussed in (Vacavant et al., 2008). If we now consider the background/foreground frontier to compute the $\mathbb{I}$-BDT, Eq. (2) implies that we compute a VD of segments, and not a classical VD of points (see Fig. 1 for an example of these diagrams computed on a simple $\mathbb{I}$-grid). Hence, a simple approach to compute the $\mathbb{I}$-CDT (presented in Algorithm 1 s to pre-compute the complete VD of the background points (Vacavant et al., 2008), and to locate foreground points in the VD. In this case, the $\mathbb{I}$-CDT computation has a $\mathcal{O}(n \log n_B)$ time complexity, where n is the total number of cells, and n_B is the number of background cells. This technique is obviously not computationally efficient for every grids, and not adapted to dense grids (Vacavant et al., 2008). To compute the $\mathbb{I}$-BDT, a similar process can be drawn, where the computation of the VD of segments can be handled in $\mathcal{O}(n_S \log^2 n_S)$, where $n_S = 4n_B$ is the total number of segments belonging to the background/foreground frontier (Karavelas, 2006). The extension of those transformations to d-D $\mathbb{I}$-grids is difficult. A VD can be computed in d-D with a $\mathcal{O}(n_B \log n_B + n_B^{\lceil d/2 \rceil})$ time complexity (thanks to a gift-wrapping approach (de Berg et al., 2000 for example). However, localizing a point in the VD is an arduous task, and an additional structure like subdivision grids (Park et al., 2005) should be constructed to handle this operation. Hence, the building of the entire VD to obtain the $\mathbb{I}$-BDT is neither computationally efficient for every $\mathbb{I}$-grids, nor easily extensible to higher dimensions.

Table 1
Differences between the two extensions of E^2DT on $\mathbb{I}$-grids.

	$\mathbb{I}$-CDT		*$\mathbb{I}$-BDT*
	Based on distance between cells centres		Based on centre-to-border distance
	Can be computed thanks to the construction of a Voronoi diagram of points		Can be computed thanks to the construction of a Voronoi diagram of segments
+	Is extensible to other non-isothetic grids (hexagonal, triangular, *etc.*)	–	Not easily extensible to these grids
–	Is not independent from background encoding	+	The result does not depend on the background representation
+	Can be computed in 2-D (but not in d-D) in linear time with a sweep line algorithm (Vacavant, 2010)	–	Not easily computable by this algorithm

This article: the two extensions can be computed with a separable approach, extensible to d-D

Algorithm 1. $\mathbb{I}$-CDT with the complete VD

```
    input: a labelled $\mathbb{I}$-grid $G$.
    output: the $\mathbb{I}$-CDT of $G$.
1   compute the Voronoi Diagram $\mathcal{V}$ of the points $\{p;\ R \in G_B\}$;
2   foreach cell $R \in G_F$ do
3       locate $p$ in $\mathcal{V}$;
4       if $p$ is the Voronoi vertex $v$ of $\mathcal{V}$ then
5           $s \leftarrow$ Voronoi site of an adjacent cell of $v$;
6       elseif $p$ belongs to a Voronoi edge $e$ in $\mathcal{V}$ then
7           $s \leftarrow$ Voronoi site of an adjacent cell of $e$;
8       else            {$p$ belongs to a Voronoi cell $c$ of $\mathcal{V}$}
9           $s \leftarrow$ Voronoi site of $c$;
10      $\mathbb{I}$-CDT $(R) \leftarrow d^2(s,p)$;
11  foreach cell $R \in G_B$ do $\mathbb{I}$-CDT $(R) \leftarrow 0$
```

In Table 1, we also mention that one of the major problem of $\mathbb{I}$-CDT is that it is strongly dependent of the background representation (Vacavant et al., 2009a). In Fig. 2, we present an example of the computation of the $\mathbb{I}$-CDT of two $\mathbb{I}$-grids where only the background cells differ. Since this definition is based on the cell centres position, the $\mathbb{I}$-CDT do not lead to the same distance map depending on the background encoding.

In a recent work (Vacavant, 2010), we have presented an efficient algorithm that permits to compute the $\mathbb{I}$-CDT with a sweep-line based approach in 2-D (Breu et al., 1995). However, this technique is not easily extensible neither to higher dimensions, nor to $\mathbb{I}$-BDT. As a new contribution, we now propose two separable algorithms that compute both $\mathbb{I}$-CDT and $\mathbb{I}$-BDT that can be extended to the d-D case.

3. Separable minimization processes for $\mathbb{I}$-CDT and $\mathbb{I}$-BDT

3.1. Separable E^2DT algorithms on regular grids

The separable algorithm introduced by Saito and Toriwaki (1994) (one phase by dimension) first computes a one-dimensional (1-D) transformation along X-axis for each row of a binary image F of size $n \times n$, which leads to a new image $G = \{g(i,j)\}$ (see Fig. 3a). Then, a minimisation process is realized along Y-axis to obtain the final image $H = \{h(i,j)\}$ (Fig. 3b): $h(i,j) = \min_y \{g(i,j)^2 + (j-y)^2;\ 1 \leqslant y \leqslant n\}$.

The main idea of the separable method of Maurer et al. (2003) is that the intersection between the complete VD of background pixels (*i.e.* sites) and a line of the grid can be easily computed, then simplified. Indeed, for a row j, the VD sites can be "deleted" by respecting three remarks mainly based on the monotonicity of the d_e distance (Maurer et al., 2003): (1) if we consider the line

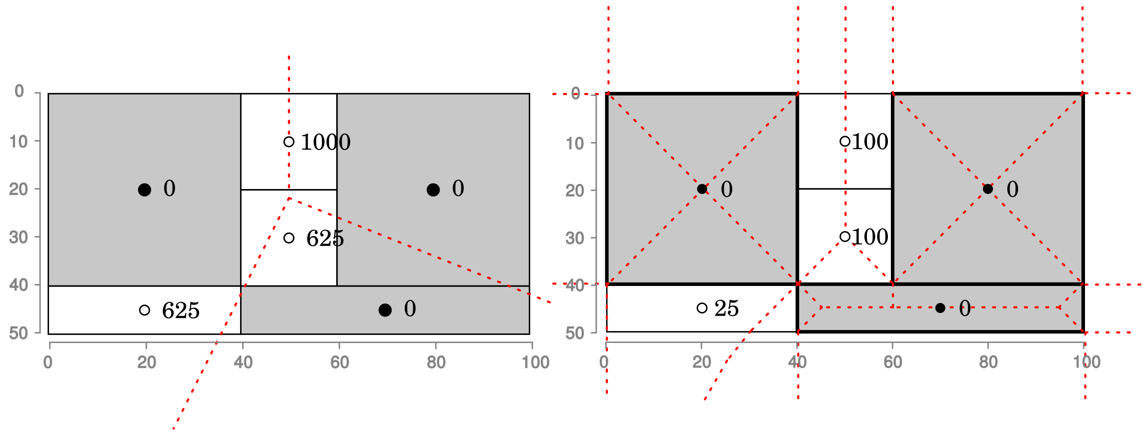

Fig. 1. Example of the VD of the background points to obtain the $\mathbb{I}$-CDT of a simple $\mathbb{I}$-grid (a) and the background/foreground frontier (dark segments) to obtain the $\mathbb{I}$-BDT (b).

$l : y = j,\ j \in \mathbb{Z}$, then we only have to keep the nearest VD sites from l (2) those sites can be ordered along the X-axis; which means that it is not necessary to keep the complete VD, but only its intersection with the line l (3) a VD site may be *hidden by* two neighbour sites, and thus not considered anymore (Fig. 4b). In this article, we show the extension of this technique on $\mathbb{I}$-grids by adapting these properties on these grids.

3.2. Separable computation of $\mathbb{I}$-CDT and $\mathbb{I}$-BDT

To develop a separable process on $\mathbb{I}$-grids, we use the *irregular matrix* $\mathbb{M}$ associated to a labeled $\mathbb{I}$-grid G introduced in (Vacavant et al., 2008). This data structure aims to organize the cells of the grid along the X- and Y-axis by adding virtual cells centres (see Fig. 5 for an example). We show in Section 5 that, even if we add points in this structure, separable processes on the irregular matrix are faster than the complete VD based approach we presented in the previous section (Algorithm 1). More precisely, the irregular matrix contains as many columns (respectively rows) as X-coordinates (Y-coordinates) in the grid. These coordinates are stored in two tables T_X and T_Y (and we denote $n_1 = |T_X|$ and $n_2 = |T_Y|$). At the intersection of two X- and Y-coordinates, a node in $\mathbb{M}$ has the same value as the cell containing it, and may represent the cell centre or not (*i.e.* this is an *extra node*, see Fig. 5a). The extra nodes are used

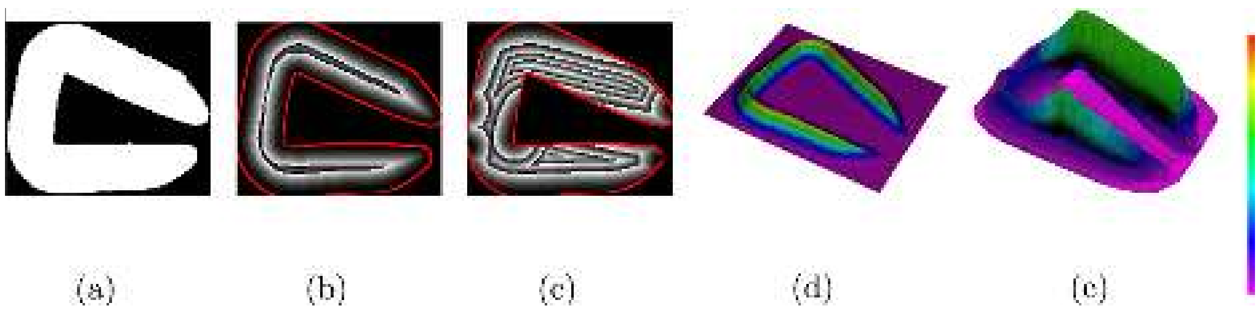

Fig. 2. The result of the $\mathbb{I}$-CDT of the complete regular grid (b) computed from the binary image (a) and an $\mathbb{I}$-grid where the foreground is regular and the background is encoded with a RLE scheme along Y (c). The distance value d of a cell is represented with a grey level $c = d$ mod 255. The contour of the object (background/foreground frontier) is recalled in each distance map (b) and (c) with a smooth curve. We also present the associated elevation map of each distance map in (d) and (e).

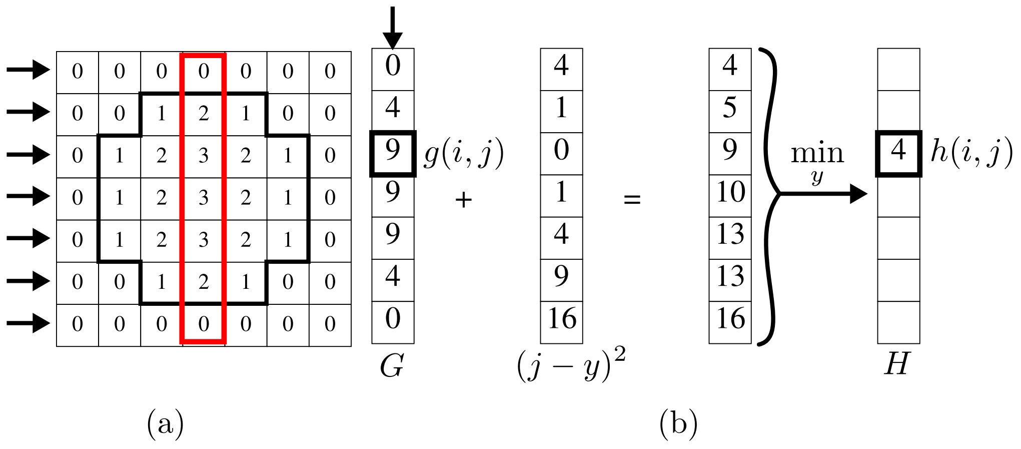

Fig. 3. Double minimization process on a small binary image. The first step of the algorithm computes G (a), representing minimal distances along X axis. If we choose one column of G, the second phase consists in computing $g(i,j)^2 + (j-y)^2$ for each pixel of this column (b). The minimization of this column lead to the computation of the E^2DT stored in H.

to propagate the distance values through the irregular matrix and then compute a correct distance value for each cell centre. To apply the 𝕀-BDT on this data structure, we also have to take into account the border and the size of the treated cells. We thus add for each node $\mathbb{M}(i,j)$ *border attributes* along *X*- and *Y*-axis that permit to propagate the distance values to cell borders through $\mathbb{M}$. For a node $\mathbb{M}(i,j)$ contained by the cell *R* in *G*, we denote respectively by $H_L(i,j)$ and $H_R(i,j)$ the attributes that represent the minimum between the distance to the left (respectively right) border of *R* along *X*, and the distance to the neighbour node at the left (right) position in $\mathbb{M}$. In the same manner, we define $H_T(i,j)$ and $H_B(i,j)$ in respect to the top (bottom) border of *R* and neighbour nodes at the top (bottom) position in $\mathbb{M}$ (see Fig. 5b). Building the irregular matrix of an 𝕀-grid *G* can be handled in $\mathcal{O}(n_1 n_2)$ time complexity. More precisely, we first scan all the cells of *G* to know the n_1 columns and the n_2 rows of $\mathbb{M}$. Then, we consider each node of $\mathbb{M}$ and we assign its background or foreground value and its border attributes.

We now present how to adapt Eq. (1) on the irregular matrix thanks to a separable minimization process:

Proposition 1 (*Separable* 𝕀-CDT). *Let* $\mathbb{M}$ *be the associated irregular matrix of the 2-D* 𝕀*-grid G. Then the* 𝕀*-CDT of G can be decomposed in two separable processes, and consists in computing the matrix* $\mathbb{M}'$ *and* $\mathbb{M}''$ *as follows:*

$$\mathbb{M}'(i,j) = \min_x \{|T_X(i) - T_X(x)| : x \in \{0,\dots,n_1-1\} \text{ and } \mathbb{M}(x,j) = 0\},$$

$$\mathbb{M}''(i,j) = \min_y \{\mathcal{F}_y(j) : y \in \{0,\dots,n_2-1\}\},$$

where $\mathcal{F}_y(j)$ *is a parabola given by:*

$$\mathcal{F}_y(j) = \mathbb{M}'(i,y)^2 + (T_Y(j) - T_Y(y))^2.$$

In this case, this transformation is equivalent to a minimization process of 2-D quadratic forms. Eq. (2) can also be adapted on the irregular matrix with a minimization process along the two axis:

Proposition 2 (*Separable* 𝕀-BDT). *Let* $\mathbb{M}$ *be the associated irregular matrix of the 2-D* 𝕀*-grid G. Then the* 𝕀*-BDT of G can be decomposed in two separable processes, and consists in computing the matrix* $\mathbb{M}'$ *and* $\mathbb{M}''$ *as follows:*

$$\mathbb{M}'(i,j) = \min_x \{\min(|T_X(i) - T_X(x) - H_R(x,j)|, |T_X(x) - T_X(i) - H_L(x,j)|) : x \in \{0,\dots,n_1-1\} \text{ and } \mathbb{M}(x,j) = 0\},$$

$$\mathbb{M}''(i,j) = \min_y \{\mathcal{G}_y(j) : y \in \{0,\dots,n_2-1\}\},$$

where $\mathcal{G}_y(j)$ *is a flattened parabola given by:*

$$\mathcal{G}_y(j) = \begin{cases} \mathbb{M}'(i,y)^2 + (T_Y(j) - T_Y(y) - H_T(i,y))^2 & \text{if } T_Y(j) - T_Y(y) > H_T(i,y), \\ \mathbb{M}'(i,y)^2 + (T_Y(y) - T_Y(j) - H_B(i,y))^2 & \text{if } T_Y(y) - T_Y(j) > H_B(i,y), \\ \mathbb{M}'(i,y)^2 & \text{otherwise.} \end{cases}$$

We prove in (Vacavant et al., 2009b) that this proposition is correct, *i.e.* this two-dimensional scheme consists in computing the

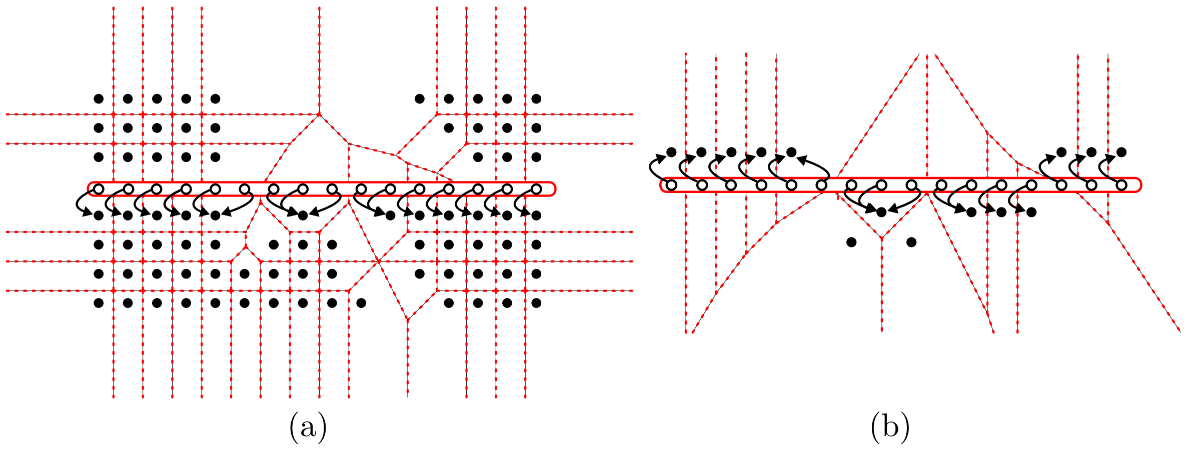

Fig. 4. In a regular grid, when we treat a row *l* with Maurer et al. algorithm, we consider the VD of background nodes like in (a).We only keep the nearest VD sites of *l* and we obtain (c) by deleting sites which associated VD cells intersect the *l* row. Arrows indicate the associated VD site of each foreground pixel of this row.

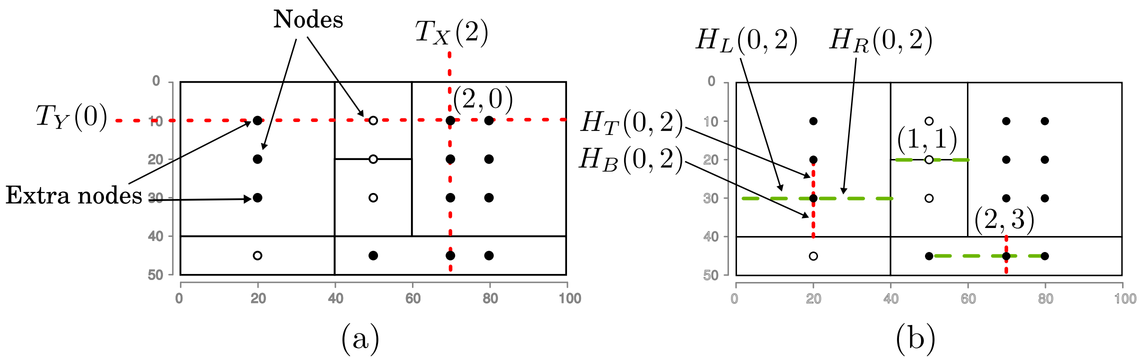

Fig. 5. Construction of the irregular matrix associated to the simple 𝕀-grid illustrated in Fig. 1. In (a), the extra node $\mathbb{M}(0,2)$ of the matrix is depicted at the intersection of the dotted lines. In (b), examples of values for border attributes are also given along *X* (dashed lines) and *Y* (dotted lines). For instance, we have $H_R(0,2) = H_L(0,2) = 20$, $H_T(0,2) = H_B(0,2) = 10$, $H_R(1,1) = H_L(1,1) = 10$, and $H_T(1,1) = H_B(1,1) = 0$ since this node coincides with a cell horizontal border. We can also notice that $H_L(2,3) \neq H_R(2,3)$ while $H_T(2,3) = H_B(2,3) = 5$.

$\mathbb{I}$-BDT given in Eq. (2). One can notice that it is possible to adapt the proof of this proposition to prove Proposition 1. This separable minimization process also permits to compute the $\mathbb{I}$-CDT (Eq. (1)), if we assign all border attributes to zero. Here, a flattened parabola (see also Fig. 6) is an *ad hoc* function composed by two half-parabolas and a constant function. It represents a constant distance value from a node $\mathbb{M}(i,j)$ to its neighbour nodes and cell borders, and then increases as a classical quadratic function beyond those limits. The computation of matrix $\mathbb{M}''$ in Proposition 2 thus consists in computing the lower envelope of a set of flattened parabolas. This operation is possible since these functions are convex (composition of convex functions), and there exist a single intersection or an infinity of intersections between two flattened parabolas. In the regular case, E^2DT algorithms based on a 2-D quadratic form minimization (Hirata, 1996; Meijster et al., 2000; Saito and Toriwaki, 1994) are linear and correct, since the intersection between two parabolas $\mathcal{F}_\alpha(y)$ and $\mathcal{F}_\beta(y)$ is clearly defined:

Proposition 3 (*Intersection between two classical parabolas (Hirata, 1996)*). *Let $\mathcal{F}_\alpha : \mathbb{R}\mapsto\mathbb{R}, y\mapsto g_\alpha^2 + (\alpha - y)^2$ and $\mathcal{F}_\beta : \mathbb{R}\mapsto\mathbb{R}, y\mapsto g_\beta^2 + (\beta - y)^2$ be two parabolas. The number of intersections is either one, or an infinity if they are coincident, i.e. when $g_\alpha = g_\beta$ and $\alpha = \beta$.*

This property is verified because parabolas are convex functions (Hirata, 1996). It implies that we can compute the lower envelope of a set of parabolas since we are able to order them.

To prove that our algorithms are correct on $\mathbb{I}$-grids for $\mathbb{I}$-BDT, we have verified that this property is preserved. The following lemma enounce the conditions so that two flattened parabolas intersect at a single point or an infinity of points:

Lemma 1 (*Intersection between two flattened parabolas*). *Let $\mathcal{G}_u$ and $\mathcal{G}_v$ be two flattened parabolas given by:*

$\mathcal{G}_u : \mathbb{R} \mapsto \mathbb{R}$

$$y \mapsto \mathcal{G}_u(y) = \begin{cases} g_u^2 + (u - y - lu_1)^2 & \text{if } u - y > lu_1, \\ g_u^2 + (y - u - lu_2)^2 & \text{if } y - u > lu_2, \\ g_v^2 \text{ otherwise} \end{cases}$$

and

$\mathcal{G}_v : \mathbb{R} \mapsto \mathbb{R}$

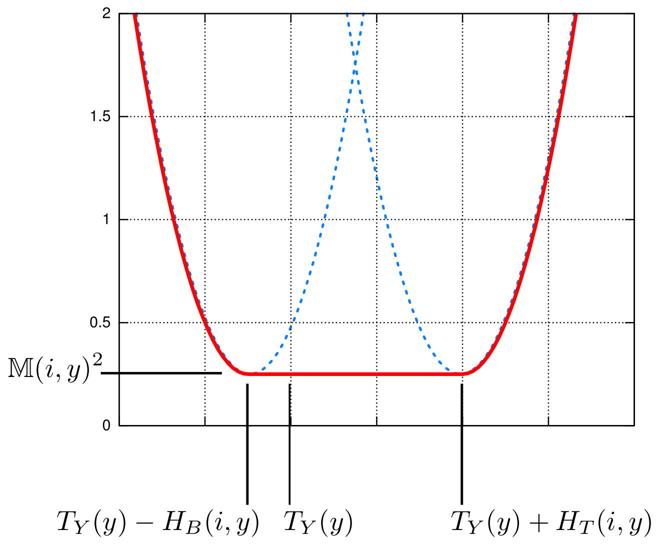

Fig. 6. Important features of a flattened parabola, composition of two half-parabolas (in dotted lines) and a constant function.

$$y \mapsto \mathcal{G}_v(y) = \begin{cases} g_v^2 + (v - y - lv_1)^2 & \text{if } v - y > lv_1, \\ g_v^2 + (y - v - lv_2)^2 & \text{if } y - v > lv_2, \\ g_v^2 \text{ otherwise} \end{cases}$$

where:

- *g_u, g_v, lu_1, lu_2, lv_1, lv_2, u and v are positive or null real numbers;*
- *if $u \geqslant v \geqslant 0$, then $u - lu_1 \geqslant v + lv_2$;*
- *if $0 \leqslant u \leqslant v$, then $u + lu_2 \leqslant v - lu_1$.*

The intersections between these parabolas is either reduced to a point, or reduced to an infinite set of points.

Notice that the elements $\mathbb{M}'(i,y), H_T(i,y), H_B(i,y)$ and $T_Y(y)$ of $\mathcal{G}_y$ in Proposition 2 respectively correspond to g_u, lu_1, lu_2, and u of $\mathcal{G}_u$. They also respect the conditions given at the end of Lemma 1. For example, let $\mathcal{G}_{y_1}$ and $\mathcal{G}_{y_2}$ be two flattened parabolas such that $y_1 \geqslant y_2$, we can easily verify that $y_1 - H_B(i,y_1) \geqslant y_2 + H_T(i,y_2)$ (see Figs. 5 and 6). The proof of this lemma and a way to compute a valid intersection point are given in (Vacavant et al., 2009b).

4. Proposed separable methods to compute *I*CDT and *I*BDT

In this section, we first present the adaptation of Saito et al. algorithm (Saito and Toriwaki, 1994), because this is a natural application of the $\mathbb{I}$-CDT and $\mathbb{I}$-BDT minimization processes we described in Propositions 1 and 2. Then, we show how to use Maurer et al. approach (Maurer et al., 2003) for the same purpose, which lead to the development of a faster algorithm.

4.1. Adaptation of Saito et al. E^2DT Algorithm on $\mathbb{I}$-grids

Algorithm 2. Separable $\mathbb{I}$-BDT inspired from Saito and Toriwaki (1994) (1/2)

```
   input: A labeled $\mathbb{I}$-grid $G$.
   output: The $\mathbb{I}$-BDT of $G$, stored in an irregular matrix $\mathbb{M}''$.
1  build the irregular matrix $\mathbb{M}$ associated to $G$;
2  for $j = 0$ to $n_2 - 1$ do                {First stage along X}
3      if $\mathbb{M}(0,j) = 0$ then $\mathbb{M}'(0,j) \leftarrow 0$;
4      else $\mathbb{M}'(0,j) \leftarrow \infty$ ;
5      for $i = 1$ to $n_1 - 1$ do
6          if $\mathbb{M}(i,j) = 0$ then $\mathbb{M}'(i,j) \leftarrow 0$;
7          else
8              if $\mathbb{M}'(i-1,j) = 0$ then
9                  $\mathbb{M}'(i,j) \leftarrow T_X(i) - T_X(i-1) - H_R(i-1,j)$;
10             else
11                 $\mathbb{M}'(i,j) \leftarrow T_X(i) - T_X(i-1) + \mathbb{M}'(i-1,j)$;
12     for $i = n_1 - 2$ to $0$ do
13         if $\mathbb{M}'(i+1,j) < \mathbb{M}'(i,j)$ then
14             if $\mathbb{M}'(i+1,j) = 0$ then
15                 $\mathbb{M}'(i,j) \leftarrow T_X(i+1) - T_X(i) - H_L(i,j)$;
16             else
17                 $\mathbb{M}'(i,j) \leftarrow T_X(i+1) - T_X(i) + \mathbb{M}'(i-1,j)$;
```

In this section, we present the complete algorithm inspired from Saito and Toriwaki (1994) (Algorithm 2, divised into two main parts), and how to modify it in order to compute the $\mathbb{I}$-CDT and $\mathbb{I}$-BDT. As we discussed in the previous section, we need to compute the intersection between parabolas to realize a correct minimization scheme of a 2-D quadratic form, conform to Propositions 1 and 2. The function $CSep^i(u,v)$ is the exact coordinate of the intersection point between two classical parabolas (Coeurjolly and Montanvert, 2007; Meijster et al., 2000): $CSep^i(u,v) = (T_Y(v)^2 - T_Y(u)^2 + \mathbb{M}'(i,v)^2 - \mathbb{M}'(i,u)^2)/(2(T_Y(v) - T_Y(u)))$. For $\mathbb{I}$-BDT, we introduce the $BSep^i(u,v)$ function thanks to the proof of Lemma 1 (see Vacavant et al., 2009b).

Algorithm 3. Separable $\mathbb{I}$-BDT inspired from Saito and Toriwaki (1994) (2/2)

```
   input: A labeled $\mathbb{I}$-grid $G$.
   output: The $\mathbb{I}$-BDT of $G$, stored in an irregular matrix $\mathbb{M}''$.
1  for $i = 0$ to $n_1 - 1$ do                                  {Second stage along Y}
       $q \leftarrow 0$; $s[0] \leftarrow 0$; $t[0] \leftarrow 0$;
2      $h_T[0] \leftarrow H_T(i,0)$; $h_B[0] \leftarrow H_B(i,0)$
3      for $j = 1$ to $n_2 - 1$ do
4          while $q \geqslant 0 \wedge \mathcal{G}_{s[q]}(t[q]) > \mathcal{G}_j(t[q])$ do   {$\mathcal{F}$ function for $\mathbb{I}$-CDT}
5              $q \leftarrow q - 1$;
6          if $q < 0$ then
7              $q \leftarrow 0$; $s[q] \leftarrow j$;
8              $h_T[q] \leftarrow H_T(i,j)$; $h_B[q] \leftarrow H_B(i,j)$;
9          else                                                 {$CSep^i$ function for $\mathbb{I}$-CDT}
10             $w \leftarrow BSep^i(s[q],j)$;
11             if $w \leqslant T_Y(n_2 - 1)$ then
12                 find the node $\mathbb{M}'(i,k)$, $k \in \{s[q],\dots,n_2-1\}$ such that $T_Y(k) > w$;
13                 $q \leftarrow q + 1$; $s[q] \leftarrow k$; $t[q] \leftarrow w$;
                   $h_T[q] \leftarrow H_T(i,j)$; $h_B[q] \leftarrow H_B(i,j)$;
14
15
16
17     for $j = n_2 - 1$ to 0 do                                 {$\mathcal{F}$ function for $\mathbb{I}$-CDT}
18         $\mathbb{M}''(i,j) \leftarrow \mathcal{G}_{s[q]}(j)$;
19         if $T_Y(j) = t[q]$ then $q \leftarrow q - 1$;
```

This function is defined by:

$$\bar{y} = T_Y(v) - H_B(i,v) - \sqrt{\mathbb{M}'(i,u)^2 - \mathbb{M}'(i,v)^2}, \tag{3}$$

$$BSep^i(u,v) = \begin{cases} \bar{y} \ \text{ if } T_Y(u) - H_B(i,u) \leqslant \bar{y} \leqslant T_Y(u) + H_T(i,u), \\ CSep(T_Y(u) - H_B(i,u), T_Y(v) - H_B(i,v)) \\ \qquad \text{if } \bar{y} \leqslant T_Y(u) - H_B(i,u) \leqslant T_Y(u) + H_T(i,u), \\ CSep(T_Y(u) + H_T(i,u), T_Y(v) - H_B(i,v)) \\ \qquad \text{if } T_Y(u) - H_B(i,u) \leqslant T_Y(u) + H_T(i,u) \leqslant \bar{y}. \end{cases}$$

In the first step, we can notice that the only difference with the regular square case (Coeurjolly and Montanvert, 2007; Saito and Toriwaki, 1994) is the computation of the distance, lines 2 to 17 (part 1/2). We have to consider in those operations the distance between the point $\mathbb{M}'(i,j)$ and its neighbors. This is indeed a double linear scan for each row, where we use H_R and H_L attributes to propagate the distance to cells borders[1]. During the second phase, we use stacks s and t, as in (Meijster et al., 2000), and two additional stacks h_T and h_B to propagate border distances through the columns of the matrix. This step is equivalent to the computation of the lower envelope of a set of flattened parabolas (see Fig. 7). The computation of w (line 11 part 2/2) only depends on the function $BSep^i(u,v)$ and then permits to find the intersection point in $\mathbb{M}'$ between two flattened parabolas (line 13 part 2/2). It is replaced by $CSep^i(u,v)$ in the case of the $\mathbb{I}$-CDT. The find command is performed with a dichotomous search through the ordered set of nodes $\{\mathbb{M}'(i,k)\}_{s[q] \leqslant k \leqslant n_2 - 1}$, and has a $\mathcal{O}(\log n_2)$ time complexity in the worst case. But in our experiments (see Section 5), we have observed that this is a fast operation, since we begin the search from the last intersection point (with index $s[q]$).

[1] If we consider an $\mathbb{I}$-CDT process, we can set these values to zero, or use the complete description given in (Vacavant et al., 2008).

4.2. Adaptation of R. Maurer et al. E^{2}DT Algorithm on $\mathbb{I}$-grids

The first stage of this method (Algorithm 3) consists in scanning along X and in initializing the distance of each node of the irregular matrix, as in Algorithm 2. In the second part of our algorithm, we build a partial VD intersected with each column i. As in the original algorithm (Maurer et al., 2003), we use two stacks storing real successive coordinates of treated sites (h), and their squared distance (g). The first loop of this function (line 7) corresponds to the deletion of hidden sites, thanks to the `hidden_by()` predicate.

Algorithm 4. Separable $\mathbb{I}$-BDT inspired from Maurer et al. (2003)

```
   input: the labeled $\mathbb{I}$-grid $G$.
   output: the $\mathbb{I}$-BDT of $G$, stored in the irregular matrix $\mathbb{M}''$.
1  build the irregular matrix $\mathbb{M}$ associated to $G$;
2                    {Process same first stage along X as Algorithm 2}
3  for $i = 0$ to $n_1 - 1$ do                                   {Second stage along Y})
4      $l \leftarrow 0$, $g \leftarrow \emptyset$, $h \leftarrow \emptyset$;
5      $f_T \leftarrow \emptyset f_B \leftarrow \emptyset$;
6      for $j = 0$ to $n_2 - 1$ do
7          if $\mathbb{M}'(i,j) \neq \infty$ then
8              while $l \geqslant 2 \wedge$ hidden_by
                   $(g[l-1], g[l], \mathbb{M}'(i,j), h[l-1], h[l], T_Y(j))$ do
9                  $l \leftarrow l - 1$;
10             $l \leftarrow l + 1$, $g[l] \leftarrow \mathbb{M}(i,j)$, $h[l] \leftarrow T_Y(j)$;
11             $f_T[l] \leftarrow H_T(i,j), f_B \leftarrow H_B(i,j)$;
12     if $(n_s \leftarrow l) = 0$ then return;
13     $l \leftarrow 1$
14     for $j = 0$ to $n_2 - 1$ do                               {$\mathcal{F}$ function for $\mathbb{I}$-CDT}
15         while $l < n_s \ \wedge \ \mathcal{G}_l(j) > \mathcal{G}_{l+1}(j)$ do $l \leftarrow l + 1$;
16         $\mathbb{M}''(i,j) \leftarrow \mathcal{G}_l(j)$;
```

Function `hidden_by`

```
   input: Y-coordinates of three points in $\mathbb{R}^2$
   denoted by $u_y$, $v_y$, $w_y$, and their squared distance
   to the line $L$: $y = r$ denoted by
   $d_e^2(u,L), d_e^2(v,L), d_e^2(w,L)$.
   output: is $v$ hidden by $u$ and $w$?
1  $a \leftarrow v_y - u_y$, $b \leftarrow w_y - v_y$, $c \leftarrow a + b$
2  return $c \times d_e^2(v,L) - b \times d_e^2(u,L) - a \times d_e^2(w,L) - abc > 0$;
```

In Algorithm 3, we also use two additional stacks, denoted by f_T and f_B to store the border attributes and update them. Thanks to these stacks, the second stage of our algorithm is achieved in linear time. By testing the value of l (line 12), we know if we have to scan again the stacks and to update the distance values of the nodes. Finally, we linearly scan the stacks to find the nearest border of the $\mathbb{M}(i,j)$ current node (line 15), and this step actually consists in considering the lower envelope of a set of flattened parabolas, as in Algorithm 2 (see also Fig. 7).

4.3. Complexity analysis

In Algorithms 2 and 3, the first operation (build the irregular matrix) is performed in $\mathcal{O}(n_1 n_2)$ time. More precisely, we first scan all the cells of G to get the n_1 rows and n_2 columns of $\mathbb{M}$. Algorithm

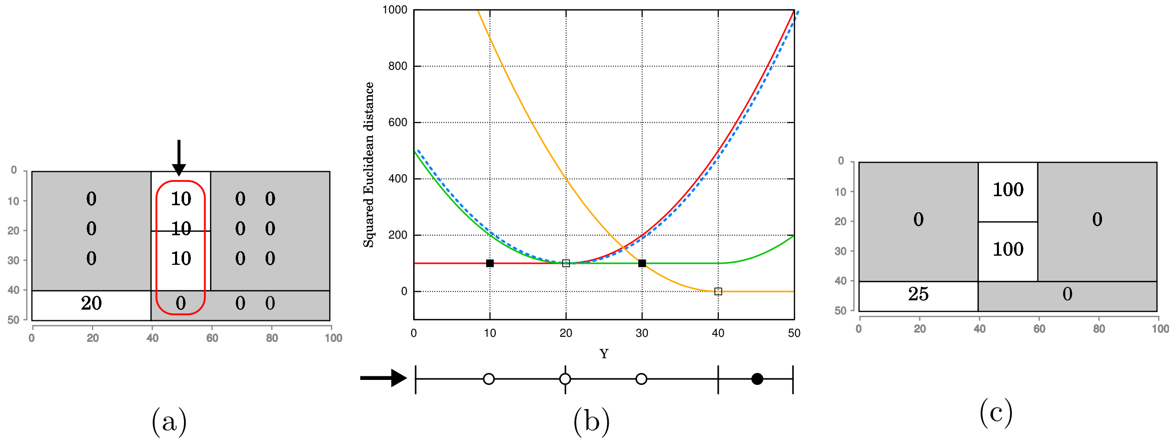

Fig. 7. For the column chosen in (a), the last phase of Algorithms 2 and 3 consists in considering the lower envelope of a set of flattened parabolas (b). At the bottom of this plot, background and foreground nodes are represented by black and white circles at the corresponding *Y*-coordinate. Cell borders are also represented (vertical dashes). Black squares represent where the cell centers are located along *Y*-axis, and the associated $\mathbb{I}$-BDT value. We give in (c) the obtained $\mathbb{I}$-BDT for this $\mathbb{I}$-grid.

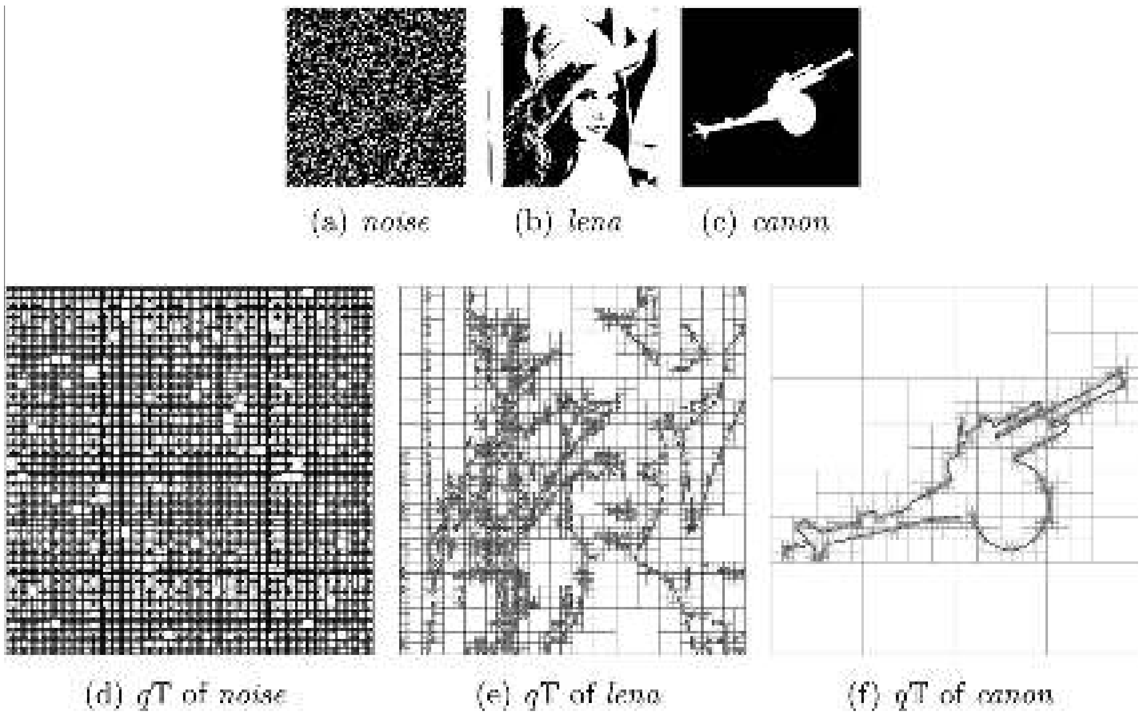

Fig. 8. For the images *noise* (100×100), *lena* (208×220), *canon* (512×512) in (a–c), we consider in our experiments the associated $\mathbb{I}$-grids (regular grid, quadtree decomposition and RLE along *Y* scheme). We have also depicted the quadtree decomposition of these images (d–f).

2 has a global time complexity in $\mathcal{O}(n_1\ n_2\ \log n_2)$. It can be easily extended to higher dimensions: the step 1 stands as an initialization step, and for each greater dimension, a mixing process, as step 2, permits to combine results obtained in the lower dimensions. For a labeled *d*-D $\mathbb{I}$-grid, which associated irregular matrix size is $n_1,\ldots,n_d$, the cost of the consecutive steps is in $\mathcal{O}(n_1\ \ldots\ n_d\ \log n_1\ \ldots\ \log n_{d-1})$.

Algorithm 3 realizes a linear $\mathbb{I}$-BDT algorithm in respect to the associated irregular matrix size, *i.e.* in $\mathcal{O}(n_1\ n_2)$ time complexity. As Algorithm 2, this method is easily extensible to higher dimensions: we still realize the first step as an initialization phase, and for each dimension $d > 1$, we combine results obtained in dimension $d - 1$. If we consider a labeled *d*-D $\mathbb{I}$-grid, which associated irregular matrix size is $n_1,\ldots,n_d$, the time complexity of our algorithm is thus in $\mathcal{O}(n_1\ \ldots\ n_d)$.

In the next section, we present experiments to show the interest of the $\mathbb{I}$-BDT and $\mathbb{I}$-CDT, and to point out that the last algorithm is a very efficient approach to compute both $\mathbb{I}$-CDT and $\mathbb{I}$-BDT of various classical $\mathbb{I}$-grids.

5. Experimental analysis

Here, we first propose to study the computation of the $\mathbb{I}$-CDT on several classical 2-D grids in imagery. We consider in our experiments three regular grids (square, rectangular and hexagonal), then two image-dependent grids, based on a quadtree decomposition and a RLE grouping scheme. Since the definition of the $\mathbb{I}$-CDT is only based on the centres of the cells, we can treat all those grids that belong to the $\mathbb{I}$-grid model, and other non-isothetic grids, like hexagonal or triangular grids. We

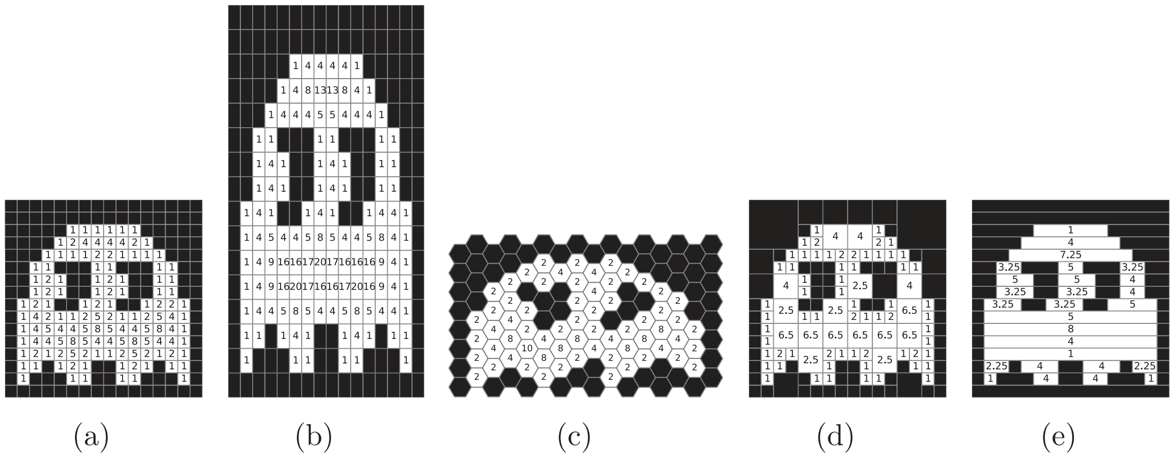

Fig. 9. Results of the 𝕀-CDT for classical 2-D grids constructed from the image *ghost*: (a) regular square grid, (b) rectangular grid, (c) hexagonal grid, (d) quadtree-based grid, and (e) a grid built with a RLE along *X*-axis.

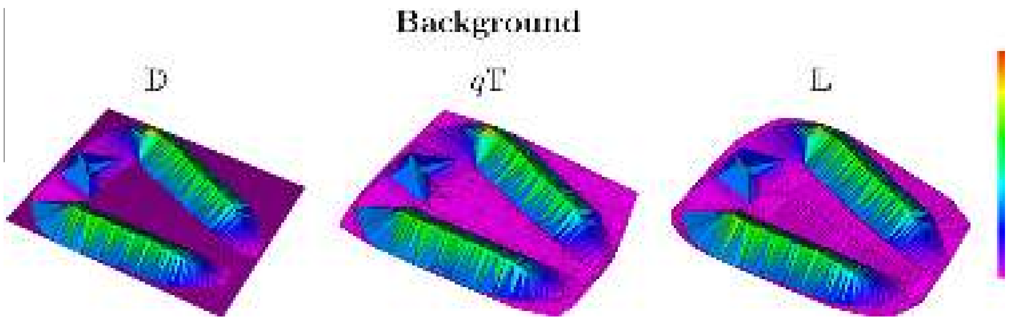

Fig. 10. Elevation maps representing the 𝕀-BDT of each 𝕀-grid. *X* and *Y* are the axis of the image, and *Z* corresponds to the distance value. The foreground is encoded with a RLE and the background (rows) of the image are digitized independently. The color palette is depicted on the right of the figure.

present the computation results of the 𝕀-CDT for those grids, based on the small image *ghost* (16×16 pixels) in Fig. 9. In this figure, the rectangular grid cells have a height two times greater than the width. We recall that 𝕀-CDT is an extension of E^2DT conform to classical E^2DT definitions on regular grids, and may be extended to such *d*-D grids.

We now propose to present the result of our 𝕀-BDT algorithm for the binary image depicted at the beginning of this article in Fig. 2, digitized in various classical 𝕀-grids used in imagery. Fig. 10 illustrates 𝕀-BDT elevation maps where the background of the original image is represented with a regular square grid ($\mathbb{D}$), a quadtree ($q\mathbb{T}$) and a RLE along $Y(\mathbb{L})$. The foreground is encoded with an RLE along *Y*-axis. We can notice in this figure that the result of the 𝕀-BDT is independent of the representation of the background. The distance values sin the foreground region are thus the same in the three elevation maps of a given column.

We now focus our interest on the execution time of the three algorithms presented in this article. Moreover, we take into account the 𝕀-CDT 2-D approach that we introduced in (Vacavant, 2010). Hence, we consider the four algorithms presented in Table 2. The first algorithm represents the simple approach we discussed in Section 2, which is hardly extensible to *d*-D treatments, and 𝕀-BDT computation. The fourth technique was recently introduced in (Vacavant, 2010) and is based on a sweep-line process (Breu et al., 1995). For this algorithm, extension to 𝕀-BDT and *d*-D case are also difficult. As its complexity is quasi optimal, it represents a reference for our experiments. Algorithm 2 is inspired from the quadratic form minimization scheme of Saito and Toriwaki (1994). Algorithm 3 is the separable transformation inspired from Maurer et al. (2003). In Fig. 8, we present the three chosen images for our experiments, and in Table 3, we show the execution times for these four algorithms, for 𝕀-grids built from three binary images. We have performed those experiments on a mobile workstation with a 2.2 GHz Intel Centrino Duo processor, and 2 Gb sRAM. We can first notice that Algorithm 3 gives good results for 𝕀-CDT, even compared to the optimal Algorithm 4. Indeed, this is the fastest one for regular square 𝕀-grids, and is very competitive for sparse 𝕀-grids (*e.g.* near one half second for $q\mathbb{T}$ and $\mathbb{L}$ based

Table 2
The three compared algorithms, and their associated time and space complexities. We also check if an algorithm is extensible to *d*-D 𝕀-grids and what kind of transformation it can perform (𝕀-CDT,𝕀-BDT).

Id.	Algorithm	Time	Space	*d*-D	𝕀-CDT	𝕀-BDT
1	Complete VD (Vacavant et al., 2008; Voronoi, 1908)	$\mathcal{O}(n \log n_B)$	$\mathcal{O}(n)$		✓	
2	From Vacavant et al. (2008) and Saito and Toriwaki (1994)	$\mathcal{O}(n_1 n_2 \log n_2)$	$\mathcal{O}(n_1 n_2)$	✓	✓	✓
3	From Vacavant et al. (2009a) and Maurer et al. (2003)	$\mathcal{O}(n_1 n_2)$	$\mathcal{O}(n_1 n_2)$	✓	✓	✓
4	From Vacavant (2010) and Breu et al. (1995)	$\mathcal{O}(n)$ best, $\mathcal{O}(n_1 n_2)$ worst	$\mathcal{O}(n)$		✓	

Table 3
We present execution times (in seconds) for each algorithm for the $\mathbb{I}$-CDT (a) and the $\mathbb{I}$-BDT (b) and for each $\mathbb{I}$-grid. Number inside parenthesis in (b) are the increasing rate in % between $\mathbb{I}$-CDT and $\mathbb{I}$-BDT execution times.

Image	Algorithm 1			Image	Algorithm 2			Image	Algorithm 3			Image	Algorithm 4		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$		$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$		$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$		$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
(*a*) $\mathbb{I}$-CDT															
Noise	0.255	0.104	0.053	*Noise*	0.037	0.077	0.044	*Noise*	0.046	0.065	0.038	*Noise*	0.112	0.055	0.016
Lena	1.413	0.145	0.081	*Lena*	0.192	0.376	0.245	*Lena*	0.185	0.166	0.135	*Lena*	0.327	0.062	0.024
Canon	36.236	0.273	0.234	*Canon*	1.678	1.322	1.316	*Canon*	1.134	0.485	0.585	*Canon*	0.675	0.068	0.031

Image	Algorithm 2			Image	Algorithm 2		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$		$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
(*b*) $\mathbb{I}$-BDT							
Noise	0.047 (27)	0.100 (29)	0.054 (23)	*Noise*	0.065 (42)	0.085(31)	0.049 (29)
Lena	0.256 (33)	0.320 (31)	0.320 (31)	*Lena*	0.258 (40)	0.170 (26)	0.170 (26)
Canon	2.248 (34)	2.107 (59)	2.020 (53)	*Canon*	1.507 (33)	0.563 (16)	0.718 (23)

on image *canon*). In the latter case (*canon*), Algorithms 1 and 4 are faster than Algorithm 3, but we recall that they are hardly extensible to higher dimensions. Algorithm 2 is interesting for dense grids, but is naturally overtaken by Algorithm 3. For the $\mathbb{I}$-BDT, Algorithms 2 and 3 suffer from an execution time increase, mainly due to the integration of the complex flattened parabola. But the last contribution remains as the fastest algorithm, and the time increasing rate is moderate for the tested grids. Large sparse $\mathbb{I}$-grids like $q\mathbb{T}$ and $\mathbb{L}$ based on *canon* are still handled in less than one second.

6. Conclusion and future works

In this article, we have recalled the two extensions of the E^2DT on $\mathbb{I}$-grids based on the cells centres ($\mathbb{I}$-CDT) and on the background/foreground frontier ($\mathbb{I}$-BDT). An application of the algorithms presented in this paper may be the computation of E^2DT in octree-based images for example. These algorithms, inspired from Maurer et al. (2003) and Saito and Toriwaki (1994), are separable; *i.e.* they are easily extensible to higher dimensions. The last algorithm based on the work of Maurer et al. (2003) is able to efficiently compute both $\mathbb{I}$-CDT and $\mathbb{I}$-BDT thanks to the irregular matrix structure, with competitive execution time and complexities.

As a future work, we first would like to study E^2DT on other generic image data structures, like (3-D and more generally d-D) point sets or triangulations for example. Moreover, we are working on developing efficient tools to compute the discrete Euclidean reduced medial axis transform (Coeurjolly and Montanvert, 2007) on $\mathbb{I}$-grids. We plan to propose centred simple forms of d-D binary objects represented within various image structures.

References

Breu, H., Gil, J., Kirkpatrick, D., Werman, M., 1995. Linear time Euclidean distance algorithms. IEEE Trans. Pattern Anal. Machine Intell. 17 (5), 529–533.
Chehadeh, Y., Coquin, D., Bolon, P., 1996. A skeletonization algorithm using chamfer distance transformation adapted to rectangular grids. In: 13th Internat. Conf. on Pattern Recognition (ICPR 1996), vol. 2. pp. 131–135.
Coeurjolly, D., Montanvert, A., 2007. Optimal separable algorithms to compute the reverse Euclidean distance transformation and discrete medial axis in arbitrary dimension. IEEE Trans. Pattern Anal. Machine Intell. 29 (3), 437–448.
Coeurjolly, D., Zerarga, L., 2006. Supercover model, digital straight line recognition and curve reconstruction on the irregular isothetic grids. Comput. Graph. 30 (1), 46–53.
de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., Janvier, 2000. Computational Geometry: Algorithms and Applications. Springer-Verlag.
Fabbri, R., Costa, L.D.F., Torelli, J.C., Bruno, O.M., 2008. 2D Euclidean distance transform algorithms: A comparative survey. ACM Comput. Surveys 40 (1), 1–44.
Fouard, C., Malandain, G., 2005. 3-D chamfer distances and norms in anisotropic grids. Image Vision Comput. 23 (2), 143–158.
Fouard, C., Strand, R., Borgefors, G., 2007. Weighted distance transforms generalized to modules and their computation on point lattices. Pattern Recognition 40 (9), 2453–2474.
Hesselink, W.H., Visser, M., Roerdink, J., 2005. Euclidean skeletons of 3D data sets in linear time by the integer medial axis transform. In: Ronse, C., Najman, L., Decencire, E. (Eds.), Computational Imaging and Vision, vol. 30. Springer-Verlag, pp. 259–268.
Hirata, T., 1996. A unified linear-time algorithm for computing distance maps. Inform. Process. Lett. 58 (3), 129–133.
Karavelas, M., 2006. Voronoi diagrams in cgal. In: 22nd European Workshop on Computational Geometry (EWCG 2006),. pp. 229–232.
Klette, R., Rosenfeld, A., 2004. Digital Geometry: Geometric Methods for Digital Picture Analysis. Morgan Kaufmann Publishers Inc..
Maurer, C.R., Qi, R., Raghavan, V., 2003. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. IEEE Trans. Pattern Anal. Machine Intell. 25 (2), 265–270.
Meijster, A., Roerdink, J., Hesselink, W.H., 2000. A general algorithm for computing distance transforms in linear time. In: Mathematical Morphology and its Applications to Image and Signal Processing, pp. 331–340.
Paglieroni, D.W., 1992. Distance transforms: Properties and machine vision applications. Graph. Models Image Process. 54 (1), 56–74.
Park, S., Lee, S.S., Kim, J., 2005. The Delaunay triangulation by grid subdivision. In: Computational Science and Its Applications, pp. 1033–1042.
Rosenfeld, A., Pfaltz, J.L., 1968. Distance functions on digital pictures. Pattern Recognition 1 (1), 33–61.
Saito, T., Toriwaki, J.I., 1994. New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications. Pattern Recognition 27 (11), 1551–1565.
Samet, H., 1990. Applications of Spatial Data Structures: Computer Graphics Image, Processing, and GIS. Addison-Wesley Longman Publishing Co., Inc.
Sintorn, I.-M., Borgefors, G., 2004. Weighted distance transforms for volume images digitized in elongated voxel grids. Pattern Recognition Lett. 25 (5), 571–580.
Vacavant, A., 2010. Fast distance transformation on two-dimensional irregular grids. Pattern Recognition 43 (10), 3348–3358.
Vacavant, A., Coeurjolly, D., Tougne, L., 2008. Distance transformation on two-dimensional irregular isothetic grids. In: 14th Internat. Conf. on Discrete Geometry for Computer Imagery (DGCI 2008). Lecture Notes in Computer Science, vol. 4292. Springer, pp. 238–249.
Vacavant, A., Coeurjolly, D., Tougne, L., 2009a. A novel algorithm for distance transformation on irregular isothetic grids. In: 15th Internat. Conf. on Discrete Geometry for Computer Imagery (DGCI 2009). Lecture Notes in Computer Science, vol. 5810. Springer, pp. 469–480.
Vacavant, A., Coeurjolly, D., Tougne, L., 2009b. A novel algorithm for distance transformation on irregular isothetic grids. Tech. Rep. RR-LIRIS-2009-009, LIRIS.
Voronoi, G., 1908. Nouvelles applications des paramtres continus + la théorie des formes quadratiques.deuxième mémoire: Recherches sur les parallélloèdres primitifs. J. Reine Angew. Math. 134, 198–287.
Vörös, J., 2001. Low-cost implementation of distance maps for path planning using matrix quadtrees and octrees. Robot. Comput.-Int. Manuf. 17 (13), 447–459.