Classification of the Distance Transformation Algorithms Under the Mathematical Morphology Approach

FRANCISCO DE ASSIS ZAMPIROLLI AND ROBERTO DE ALENCAR LOTUFO

FEEC-Faculdade de Engenharia Elétrica e de Computação
6101 - 13083-970 - Campinas, SP, Brasil
{fz,lotufo}@dca.fee.unicamp.br

Abstract. The Distance Transform (DT) is a morphological erosion of the binary image by a given structuring function, that dictates the distance metric in the transformation. There are many known algorithms and structuring function decompositions to efficiently implement a morphological erosion. Most of the erosion algorithms are classified as parallel, sequential raster (and anti-raster), and propagation. Based on these classification and decomposition, we review and classified most of the DT algorithms reported in the literature. As a result of this study, we have contributed not only to better classify and understand the diversity of the DT algorithms in the literature, but also to create a collection of efficient erosion algorithms suitable to different computer architectures.

Keywords – Mathematical morphology, image processing, gray-scale morphology, algorithms, distance transformation, erosion, structuring function decomposition.

1 Introduction

The *Distance Transformation* (DT) is one of the classical operators in image processing and can be defined as follows. Given a binary image with values $\{0, k\}$, where $k \neq 0$ represents the object and 0 represents the background, the DT returns gray-scale image, where the value of each pixel represents the minimum distance to the background. In spite of this definition be simple, there are several DT algorithms, however they share common concepts that allow their classification. We can implement the DT exploring the different kinds of metric: *City-Block, Chessboard, Octagonal, Chamfer* and *Euclidean* [Bor86]. We classify the DT algorithms in three categories depending on the order in which the pixels are scanned [SM92]: parallel; *raster* and *anti-raster* order [RP66, Bor86]; and propagation [Vin92].

The main goal of this work is to model the DT algorithms as an erosion as introduced by Shih and Mitchell [SM92]. Thus, for each kind of erosion implementation, parallel, sequential and propagation, and for each kind of the structuring function decomposition, it is possible to classify most of the algorithms reported in the literature.

Vincent [Vin92] made a classification of the DT, however he did not analyze the fact that the DT algorithms are morphological erosions. Rosenfeld and Pfaltz [RP66] defined a first sequential algorithm for the DT. We rewrite of the original form of this algorithm as a sequential erosion. Barrera and Hirata [BHJ97] rewrote the reconstruction algorithm using queue. They used the queue algorithm for the dilation through the border concept. We also use this border concept to define algorithms of DT using propagation erosions.

The structure of this article is the following: in the next section we present some basic concepts about mathematical morphology, metric and DT using erosion. In the third section a classification of the erosion algorithm is presented. Then, we introduce a classification for DT in the section four. Finally, in section five we describe the conclusion and future directions.

2 Basic concepts

In this section we introduce some necessary basic concepts for this work. In the first subsection we describe the mathematical morphology, the second subsection we define some metric spaces and the third subsection we define DT through the morphological erosion.

2.1 Mathematical morphology

An elegant form to solve image processing problems is the utilization of a consistent theoretical base. One of these theories is the *mathematical morphology*, created in the 60's by Jean Serra and George Matheron at the *École Nationale Superiéure des Mines of Paris*, in Fontainebleau, France. In this theory, we do transformations between lattices, which are called of the *morphological operator*. In the mathematical morphology, we have four classes of basic operators: dilations, erosions, anti–dilations and anti–erosions, which are called *elementary lattice operators*. Banon and Barrera [BB93] proved that all of the morphological operators can be obtained from combinations of these elementary lattice operators, together with the union and intersection operations. Besides, when the lattices own a sup-generating family, these operators can be characterized by *structuring functions*.

Let $\mathbf{Z}$ be the integer numbers set, $\mathbf{E} \subset \mathbf{Z}^2$ the domain of the image and $K = [0, k] \subset \mathbf{Z}$ an integer numbers interval representing the possible gray-scale of the image. The translation invariant erosion operator in gray-scale, $\varepsilon_b : K^{\mathbf{E}} \to K^{\mathbf{E}}$ ($K^{\mathbf{E}}$, it reads set of the functions of $\mathbf{E}$ in K), is defined as [Ser82, SM92]:

$$\varepsilon_b(f)(x) = \min\{f(y) \dot{-} b(y - x) : y \in (B + x) \cap \mathbf{E}\}, \quad (1)$$

where $f \in K^{\mathbf{E}}$, $x \in \mathbf{E}$, $B \in \mathcal{P}(\mathbf{Z}^2)$ ($\mathcal{P}(\mathbf{E})$ is the set of the parts of $\mathbf{E}$ and B is called *structuring element*), $B + x = \{y + x,\ y \in B\}$ (translation of B by x) and b is a *structuring function* defined on B with $b : B \to \mathbf{Z}$. When the b elements are all zeros, b is called *flat structuring function*, otherwise, *non-flat*. Let $v \in \mathbf{Z}$ be, we define $t \to t \dot{-} v$ in K by

$$\begin{cases} 0 \dot{-} v = 0 & if\ t < k\ and\ t - v \leq 0; \\ t \dot{-} v = t - v & if\ t < k\ and\ 0 \leq t - v \leq k; \\ t \dot{-} v = k & if\ t < k\ and\ t - v > k; \\ t \dot{-} v = k & \forall v \in \mathbf{Z}. \end{cases}$$

Using this erosion operator we can obtain the DT, as we will see in the section 2.3.

2.2 Metric

Let x and $y \in \mathbf{Z}^2$. $d(x, y)$ is a *distance* between x and y, if:

(i) $d(x, y) = d(y, x)$;

(ii) $d(x, y) \geq 0$;

(iii) $d(x, x) = 0$.

If iv and v, below, also are satisfied, then d it is a *metric*.

(iv) $d(x, y) = 0 \iff x = y$;

(v) $d(x, y) \leq d(x, z) + d(z, y)$.

Some kinds of metric for $d(x, y)$, where $x = (x_1, x_2) \in \mathbf{Z}^2$ and $y = (y_1, y_2) \in \mathbf{Z}^2$, are presented as follows:

City-Block: $d_4(x, y) = |x_1 - y_1| + |x_2 - y_2|$;

Chessboard: $d_8(x, y) = \max\{|x_1 - y_1|, |x_2 - y_2|\}$;

Euclidean: $d_E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

In our studies we work with the gray-scale as being a subset of the integer numbers $\mathbf{Z}$. For the Euclidean metric, we work with the squared Euclidean distance $(d_E)^2 \in \mathbf{Z}$, to stay with integer images.

We define *distance function of a pixel x to a set X*, as:

$$d(x, X) = \min\{d(x, y) : y \in X\}.$$

The *distance function* (denoted $\Psi_d(f)$ or simply DT) is defined as:

$$\Psi_d(f)(x) = d(x, \{y \in E : f(y) = 0\}),$$

that attributes for each pixel of the object the minimum distance value to the background.

2.3 DT using erosions

Shih and Mitchell were the first to show that the DT can be obtained by the morphological erosion using a structuring function b_G applied to a gray-scale image f of values 0 and k:

$$\Psi_d(f) = \varepsilon_{b_G}(f).$$

The radius of b_G and the value k must be larger than the largest distance of the object in f. The origin of b_G is zero with decreasing negative values from the origin. The shape of b_G depends on the metric used in the DT.

A further requirement of this DT operator is the *idempotency*, i.e., if we apply the erosion by b_G again, the result remains the same:

$$\varepsilon_{b_G}(\varepsilon_{b_G}(f)) = \varepsilon_{b_G}(f).$$

For example, the Figure 1 show b_G for the metric *City-Block*, *Chessboard*, *Octagonal* and *square of the Euclidean*[1], considering that the origin is in boldface and that the largest distance in the resultant image is 2.

In Figure 2b is the illustration of the erosion of the binary image f (Figure 2a) by b_G, using the Euclidean metric. As the largest distance inside the object is 2, it is enough to work with b_{G_E} of the Figure 1d.

3 Erosion efficient algorithms

The direct application of the erosion by a structuring function of size proportional to the larger object of an image is inefficient. It is possible to decompose the structuring function b_G to obtain faster implementation.

There are two main classes of algorithms in the image processing literature since the decade of 60. The *parallel algorithms* (or *iterative*) and the *sequential algorithms* (or *recursive*). Inspired mainly on the works of Rosenfeld and Pfaltz [RP66, RP68], Shih and Mitchell [SM92], Vincent [Vin92] and Barrera and Hirata [BHJ97], we propose a classification for the erosion and for DT algorithms.

[1] We are working with the squared Euclidean metric to keep the gray-scale values as integers.

		-2		
	-2	-1	-2	
-2	-1	**0**	-1	-2
	-2	-1	-2	
		-2		

(a) b_{G_4}

-2	-2	-2	-2	-2
-2	-1	-1	-1	-2
-2	-1	**0**	-1	-2
-2	-1	-1	-1	-2
-2	-2	-2	-2	-2

(b) b_{G_8}

	-2	-2	-2	
-2	-1	-1	-1	-2
-2	-1	**0**	-1	-2
-2	-1	-1	-1	-2
	-2	-2	-2	

(c) b_{G_O}

		-4		
	-2	-1	-2	
-4	-1	**0**	-1	-4
	-2	-1	-2	
		-4		

(d) b_{G_E}

Figure 1: Kinds of metric for structuring function b_G: (a) *City-Block*, (b) *Chessboard*, (c) *Octagonal* and (d) *Euclidean*.

3.1 Parallel erosion

In the parallel algorithms, the pixels are processed independently of the sweeping order in the image, depending just on the pixels values of the input image f and of the neighborhood b.

Below is a pseudo-code of the parallel erosion:

Function $g = eroPar(f,b)$
for all $x \in \mathbf{E}$ in parallel
  $g(x) = \min\{f(y) \dot{-} b(y-x) : y \in (B+x) \cap \mathbf{E}\}$;

This pseudo-code is simple, however it is slow if we work directly with b_G. To justify this affirmation, consider the next definitions:

Let $B_i \in \mathcal{P}(\mathbf{E})$ and $b_i : B_i \to \mathbf{Z}$, where $i = 1, \dots k$, then we define the *gray-scale Minkowski Addiction* of b_i by b_j [Ser82, SM92], as: $\forall x \in B_i \oplus B_j$,

$$(b_i \oplus b_j)(x) = \max\{b_i(y) + b_j(x-y) : y \in (\breve{B}_j + x)\},$$

where $j = 1, \dots k$, $\breve{B}_j = \{x \in \mathbf{E} : -x \in B_j\}$ (reflection of B_j) and $B_i \oplus B_j$ is the Set Minkowski Addiction[2] (binary images).

We can generalize the Minkowski addiction in gray-scale, doing:

$$b_G = b_1 \oplus \cdots \oplus b_k,$$

where $\{b_1, \cdots, b_k\}$ are the elements that decompose b_G. When exists a b such that,

$$b_G = \underbrace{b \oplus \cdots \oplus b}_{k\ times},$$

[2]Note that we use the same symbol for Minkowski addiction over binary images (represented by capital letters) and over gray-scale images (represented by minuscule letters).

0	0	0	0	0	0	0	0	0
0	0	0	k	k	k	0	0	0
0	0	k	k	k	k	k	0	0
0	k	k	k	k	k	k	k	0
0	k	k	k	0	k	k	k	0
0	k	k	k	k	k	k	k	0
0	0	k	k	k	k	k	0	0
0	0	0	k	k	k	0	0	0
0	0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	2	4	2	1	0	0
0	1	2	2	1	2	2	1	0
0	1	4	1	0	1	4	1	0
0	1	2	2	1	2	2	1	0
0	0	1	2	4	2	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0

(b)

Figure 2: (a) Input image f and (b) $DT = \varepsilon_{b_G}(f)$.

then we just write $b_G = kb$.

An erosion property is [Ser82, SM92]:

$$\varepsilon_{b_G}(f) = \varepsilon_{b_k}(\cdots(\varepsilon_{b_1}(f))\cdots). \quad (2)$$

We can notice that it is more efficient to work with the decomposition of b_G in the parallel erosion. This also applies for the raster and propagation erosions, as we will see in the next sections.

For example, let b_i be of dimension 3×3 and f of dimension $n \times n$. Then $\varepsilon_{b_G}(f)$ requires $(3k-1)^2n^2$ memory accesses, while $\varepsilon_{b_k}(\cdots(\varepsilon_{b_1}(f))\cdots)$ requires $9kn^2$ accesses. The value k is, in the worse case, the value of the diagonal of the rectangle $\mathbf{E}$.

Figure 3a, b and c, show the structuring functions examples, when b_i are all equal in the decomposition of $b_G = kb$ for the metrics *City-Block*, *Chessboard* and *Octagonal*, respectively.

Other kinds of decompositions for b_G are related to *Chamfer 3-4* and *Chamfer 5-7-11* metric, shown in Figures 4a and 4b, respectively [Bor86]. The Chamfer metric was created to approximate the Euclidean metric.

Huang and Mitchell [HM94] have shown that the Euclidean metric function structuring b_E can be decomposed in distinct 3×3, b_i (see Figure 5).

	-1	
-1	**0**	-1
	-1	

(a) b_4

-1	-1	-1
-1	**0**	-1
-1	-1	-1

(b) b_8

	-2		-2	
-2	-1	-1	-1	-2
	-1	**0**	-1	
-2	-1	-1	-1	-2
	-2		-2	

(c) b_O

Figure 3: Elementary structuring function used in the decompositions of the metrics: (a) *City-Block*, (b) *Chessboard* and (c) *Octagonal.*

-4	-3	-4
-3	**0**	-3
-4	-3	-4

(a) b_{3-4}

	-11		-11	
-11	-7	-5	-7	-11
	-5	**0**	-5	
-11	-7	-5	-7	-11
	-11		-11	

(b) b_{5-7-11}

Figure 4: Elementary structuring function used in the decompositions of the metrics: (a) Chamfer 3-4 and (b) Chamfer 5-7-11.

3.2 Sequential raster erosion

Unlike the parallel algorithm, in the sequential case, as the pixel are processed, the new computed pixel, rather than the original value, is used in processing any succeeding pixels which have it as neighbor.

The sequential algorithms are classified based on the order of the pixel accesses of the input image. The two most common orders are raster (and anti–raster) and propagation. For the sequential erosion, we call sequential raster erosion and propagation erosion corresponding to these type of pixel accesses.

Consider an image f with domain $\mathbf{E}$ of dimensions $m \times n = |\mathbf{E}|$. A pixel in $\mathbf{E}$ is denoted by the ordered pair (i, j), where $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$.

We define a sweeping in f in the *raster* order as a visit to the pixels of f from left to right and of the top to bottom, i.e., $\{(0,0), (0,1), \cdots, (0, n-1), (1,0), (1,1), \cdots, (1, n-1), \cdots, (m-1, 0), (m-1, 1), \cdots (m-1, n-1)\}$. We call this sequence $S^+ = \{0, \cdots, mn-1\}$. The sweeping in the *anti–raster* order in f is accomplished in the inverse direction (of the right to left and of the bottom to top), i.e., $\{(m-1, n-1), (m-1, n-2), \cdots, (m-1, 0), \cdots, (0, n-1), (0, n-2), \cdots, (0,0)\}$. We call this sequence of $S^- = \{mn-1, \cdots, 0\}$.

There is a bijector function $\ell^+ : \mathbf{E} \to S^+$, defined for $\ell^+(i,j) = in + j$, where $(i,j) \in \mathbf{E}$ and n is the width. An inverse of ℓ^+ is given by $(\ell^+)^{-1}(r) = (\lfloor r/n \rfloor, r\%n)$, where $r \in S^+$, $\lfloor r/n \rfloor$ is the integer division and $r\%n$ is the rest of the division r by n, respectively.

$-4i+2$	$-2i+1$	$-4i+2$
$-2i+1$	**0**	$-2i+1$
$-4i+2$	$-2i+1$	$-4i+2$

Figure 5: $(b_E)_i$ variable as the Equation 2.

Given $B \in \mathcal{P}(\mathbf{E})$, let B^+ (B^-) be a neighborhood for the raster order (anti–raster). Considering the center of B as the center of an axis of Cartesian coordinate $(i,j) \in \mathbf{E}$, where i is the abscissa (in downward position) and j is the ordinate (in rightward position), we have: $B^+ = \{(i,j) \in B \;/\; j < 0 \text{ or if } j = 0 \text{ then } i \leq 0\}$. Similarly $B^- = \{(i,j) \in B \;/\; j > 0 \text{ or if } j = 0 \text{ then } i \geq 0\}$. Note that $B^+ \oplus B^- = B$. This means that any structuring function can be decomposed in its raster and anti-raster structuring element.

Similarly the same decomposition can be applied to the function b: $b^+ : B^+ \to \mathbf{Z}$ and $b^- : B^- \to \mathbf{Z}$. An example of this decomposition is shown in Figure 6.

-4	-3	-4
-3	**0**	-3
-4	-3	-4

b_{3-4}

-4	-3	-4
-3	**0**	

b^+_{3-4}

	0	-3
-4	-3	-4

b^-_{3-4}

Figure 6: Decomposition of b_{3-4} in b^+_{3-4} and b^-_{3-4}.

Let $\varepsilon^+_{b^+}(f)$ be the sequential erosion in the raster order, as defined in the Equation 1, however placing the partial results in the own function f. Being more precise: for $x \in \mathbf{E}$ in the raster order,

$$\varepsilon^+_{b^+}(f)(x) = \min\{\varepsilon^+_{b^+}(f)(y) \dot{-} b(y-x) : y \in (B^+ + x) \cap \mathbf{E}\}. \quad (3)$$

Similarly, we define $\varepsilon^-_{b^-}(f)(x)$, as being the sequential erosion of f in the anti–raster order as: for $x \in \mathbf{E}$ in the anti–raster order,

$$\varepsilon^-_{b^-}(f)(x) = \min\{\varepsilon^-_{b^-}(f)(y) \dot{-} b(y-x) : y \in (B^- + x) \cap \mathbf{E}\}. \quad (4)$$

An important result, from Rosenfeld and Pfaltz [RP66], says that if we apply a sequential operator with local neighborhood on an image f it is equivalent to apply a sequence of $|\mathbf{E}|$ parallel operators with the same local neighborhood over f.

An application of this result, doing restrictions in b, see Figure 7, where $2p < q < p \leq 0$ [Bor86], is: for $x \in \mathbf{E}$ in the raster order,

$$\varepsilon^+_{b^+}(f) = \varepsilon_{mnb^+}(f).$$

q	p	q
p	**0**	p
q	p	q

b

Figure 7: Decomposition b for the equality 5.

Similarly, for $x \in \mathbf{E}$ in the anti–raster order,

$$\varepsilon_{b^-}^-(f) = \varepsilon_{mnb^-}(f).$$

We can also obtain a parallel operator through sequential operators without changing the implementation, for example: Let $k \geq mn$,

$$\varepsilon_{b_G} = \varepsilon_{kb} = \varepsilon_{k(b^+ \oplus b^-)} = \varepsilon_{kb^+ \oplus kb^-} = \varepsilon_{kb^-}(\varepsilon_{kb^+}(f)) = \varepsilon_{b^-}^-(\varepsilon_{b^+}^+(f)), \quad (5)$$

where $f \in [0,k]^{\mathbf{E}}$.

So if there is b such that $b_G = kb$ then the erosion by b_G can be implemented by the sequential raster and anti-raster algorithm:

```
Function eroSeq(f, b)  {f is input and output parameter}
for all x ∈ E in the raster order
  f(x) = min{f(y) ∸ b(y − x) : y ∈ (B⁻ + x) ∩ E};
for all x ∈ E in the anti–raster order
  f(x) = min{f(y) ∸ b(y − x) : y ∈ (B⁺ + x) ∩ E};
```

3.3 Propagation erosion

The general idea of the propagation algorithms is to process only the pixels that can be modified by the operator. These pixels coordinates are usually stored in a set and are called front or border of f.

Let $f \in K^{\mathbf{E}}$ and $B \in \mathcal{P}(\mathbf{Z}^2)$ with origin. The propagation border f of the erosion by $b \in \mathbf{Z}^B$ is the subset ∂f_b [BHJ97], where

$$\partial f_b = \{x \in \mathbf{E} : \exists y \in B + x, f(y) > f(x) \dot{-} b(x-y)\}.$$

Thus, the propagation erosion of the f by $b \in \mathbf{Z}^{\mathbf{B}}$ is defined as: $\forall x \in \mathbf{E}$,

$$\varepsilon_b^p(f)(y) = \begin{cases} \min\{f(x) \dot{-} b(x-y) : \\ x \in (B+y) \cap \partial f_b\} \ \ and \\ \\ f(x), \quad \text{otherwise.} \end{cases}$$

Below is a pseudo-code that returns the border of f using the neighborhood defined by the structuring function b. This border is placed in the set ∂f_b.

```
Function ∂f_b = front(f, b)
for all x ∈ E
    ∂f_b = {x : ∃ y ∈ (B + x) ∩ E, f(y) > f(x)∸b(x − y)};
```

Below is the code for the propagation erosion, using two sets, ∂f_b and ∂g_b. At the same time as the erosion in calculated, the new border, ∂g_b, is also calculated.

```
Function [g, ∂g_b] = eroPro(f, b, ∂f_b)
{g and ∂g_b are output parameters}
g = f;
for all x ∈ ∂f_b
  for all y ∈ (B + x) ∩ E
    if g(y) > f(x) ∸ b(x − y)
      g(y) = f(x) ∸ b(x − y);
      set_in(∂g_b, y);
```

where $set\_in(\partial g_b, y)$ is the function to insert y into the set ∂g_b.

This algorithm is very efficient for the situation where the erosion is iterated as it is the case of the DT and the morphological reconstruction operator.

As with the case of parallel erosion, the decomposition of b_G using the equation 2 is also valid for propagation erosion.

4 Classification of the main DT algorithms

Based on the three implementation of the erosion: parallel, raster and propagation, the main DT algorithms are rewritten using the framework described in the previous section.

4.1 Parallel DT

Bellow is the pseudo-code of the parallel DT, using the decomposition of b_G in $b_1, b_2, \cdots, b_k$, based on the Equation 2:

```
Function g = distPar(f, b_1, ···, b_k)
for each b_i, i = 1, , ··· k
  g = eroPar(f, b_i);
  f = g;
```

The first parallel DT algorithm was introduced by Rosenfeld and Pfaltz and published in 1968 [RP68]. Other work that implement the DT in parallel is of Borgefors [Bor86]. In both of them, the input image assumes zero and infinite values. The algorithm is described below:

$$v_{i,j}^m = \min_{(k,l) \in mask} (v_{i+k,j+l}^{m-1} + c(k,l)), \text{ until stability}, \quad (6)$$

where $v_{i,j}^m$ is the pixel value at position (i,j) of the image in the iteration m, (k,l) is the position in the $mask$, and $c(k,l)$ is the mask value at (k,l) $(c(0,0) = 0)$.

We can notice that the Equations 1 and 2 are equivalent to 6 resulting in the same DT, considering $b = b_i = -mask, i = 1, \cdots, k$. This conclusion is described in more detail in Shih and Mitchell [SM92].

To better analyze these two equations consider $x = (i,j)$ and $y = (i+k, j+l)$. Thus the Equation 6 would be $v^m(x) = \min\{v^{m-1}(y) \dot{-} b(y-x) : y \in B + x\}$, where $x \in \mathbf{E}$. Note that, making $v^0 = f$ and $v^1 = \varepsilon_b(f)$, we have the erosion definition of Equation 1.

4.2 Sequential raster DT

Below is the pseudo-code of the sequential raster DT, using the decomposition of b_G in kb. Note that it is not possible to use a decomposition with different b_i, as in the parallel case:

Function g = *distSeq(f, b)*
eroSeq(f, b);
$g = f$;

The first sequential raster DT algorithm was also reported by Rosenfeld and Pfaltz and published in 1966 [RP66]. In their algorithm, the input image contains only values zero and one, and the set of pixels with value zero is nonempty. The algorithm is,

$$f_1(a_{i,j}) = \begin{cases} 0, & if\ a_{i,j} = 0, \\ \min(a_{i-1,j}+1, a_{i,j-1}+1), & if \\ (i,j) \neq (1,1)\ and\ a_{i,j} = 1, & \\ m+n, & if \\ (i,j) = (1,1)\ and\ a_{i,j} = 1,\ and & \end{cases} \tag{7}$$

$$f_2(a_{i,j}) = \min(a_{i,j}, a_{i+1,j}+1, a_{i,j+1}+1),$$

where $a_{i,j}$ is the pixel value at position (i,j) in an image with m rows and n columns. The values $a_{i,j}$ outside the image are not defined. f_1 is applied in the raster order and, over the result, f_2 is applied in the anti–raster order. This DT uses the *City-Block* metric. If $a_{1,1} = 1$, the algorithm makes $f_1(a_{1,1}) = m + n$, that is the largest distance in the image. This algorithm step could be eliminated if we assume the input image with values zero and ∞. Besides, the first line could also be eliminated if we include the first line in the min function. Thus, $f_1(a_{i,j}) = \min(a_{i,j}, a_{i-1,j}+1, a_{i,j-1}+1)$ could substitute the step f_1 of the above algorithm.

Making b^+ and b^- the *City-Block* raster and anti–raster order function decomposition, respectively, with center in the value zero in boldface, as shown in Figure 8, we can rewrite the Equation 7 as:

$$f_1(a_{i,j}) = \min\{a_{i,j} - b^+_{0,0}, a_{i-1,j} - b^+_{-1,0}, a_{i,j-1} - b^+_{0,-1}\},$$
$$f_2(a_{i,j}) = \min\{a_{i,j} - b^-_{0,0}, a_{i+1,j} - b^-_{1,0}, a_{i,j+1} - b^-_{0,1}\}.$$

	-1			**0**	-1
-1	**0**			-1	

b_4^+ $\qquad$ b_4^-

Figure 8: *City-Block* raster b_4^+ and anti–raster b_4^- decompositions.

If $x = (i,j)$ and $f(x) = a_{i,j}$, then $f_1(a_{i,j}) = \varepsilon^+_{b^+}(f)(x)$ and $f_2(a_{i,j}) = \varepsilon^-_{b^-}(f)(x)$, as defined in the Equations 3 and 4, respectively. Rosenfeld and Pfaltz algorithm is then equivalent to our sequential raster DT $distSeq$ [RP66]. Borgefors has also implemented a sequential algorithm for DT in [Bor86], which is also similar to $distSeq$.

4.3 Propagation DT

Vincent [Vin92] implement a propagation DT. His algorithm based on the queue data structure is presented below:

```
Function distVinc(f, B)
{f is input (f ∈ {0,1}^E) and output parameter}
for all x ∈ E
   if f(x) = 1 and ∃y ∈ (B + x) ∩ E: f(y) = 0
      FIFO_add(x);
      f(x) = 2;
while FIFO_empty() = false
   x = FIFO_first();
   for all y ∈ (B + x) ∩ E
      if f(y) = 1
         f(y) = f(x) + 1;
         FIFO_add(y);
```

where $FIFO\_add(x)$, $FIFO\_empty()$ and $FIFO\_first()$ are the primitive of queue manipulation that adds an element x in a queue, verifies if the queue is empty and return the first queued element, respectively.

This algorithm can be rewritten as follows.

```
Function distAux(f, B)
{f is input (f ∈ {0,k}^E) and output parameter}
for all x ∈ E
   for all y ∈ (B + x) ∩ E
      if f(y) > f(x) + 1
         FIFO_add(x);
         break;
while FIFO_empty() = false
   x = FIFO_first();
   for all y ∈ (B + x) ∩ E
      if f(y) > f(x) + 1
         f(y) = f(x) + 1;
         FIFO_add(y);
```

The first **for all** places in the queue all of the border points of the image f. This loop can be substituted by the function $front$, which places the border points in the set ∂f_b as in the code of the algorithm $distPro$ shown below. In the **while** loop, all the border points (from the queue) are used to compute the erosion of their neighbor pixels and at the same time a new border point is computed and inserted into the queue. This process is a particular case of the $eroPro$ algorithm of the previous section, considering a non-flat structuring function. It is not necessary to use the queue, but two sets, being swapped at each iteration.

Below is the pseudo-code of the propagation DT, that is the Vincent algorithm generalization:

Function $f = distPro(f, b)$
$\partial f_b = front(f, b);$
while $\partial f_b \neq \emptyset$
  $[f, \partial f_b] = eroPro(f, b, \partial f_b);$

5 Conclusion

In this work we have presented a new classification of the Distance Transformation (DT), inspired in the morphological erosions by structuring functions. We mapped the main DT algorithms reported in the literature using morphological erosions. We have highlighted the structuring functions decomposition used in the parallel, sequential raster and propagation erosion algorithms. Table 1 summarizes classification of the main DT algorithms, in parallel, raster and propagation algorithms; and their structuring function decomposition $b_4, b_8, b_O, b_{3-4}, b_{5-7-11}$ and b_E.

When it is possible to decompose the structuring functions and these decompositions satisfy some conditions, the sequential raster algorithms present better computational performance.

In the future, we will study the coding of the Euclidean DT and of other operators, for example, the morphological reconstruction, exploring the different algorithms presented in this work. We will study further the composition of generic structuring functions in the sequential raster algorithm.

6 Acknowledgments

The first author is supported by FAPESP, process: 98 / 06641–6.

References

[BB93] G. J. F. Banon and J. Barrera. Decomposition of mappings between complete lattices by mathematical morphology, Part I: general lattices. *Signal Processing*, 30,:299–327, 1993,.

[BHJ97] Junior Barrera and Roberto Hirata Junior. Fast algorithms to compute the elementary operators of mathematical morphology. In *SIBGRAPI'97 - X Brazilian Symposium of Computer Graphic and Image Processing*, pages 163–170, São Paulo-SP, October 1997. IEEE Computer Society. URLib repository: dpi.inpe.br/banon/1997/10.03.09.49.

[Bor86] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34:344–371, 1986.

[HM94] C. T. Huang and O. R. Mitchell. A Euclidean distance transform using grayscale morphology decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:443–448, 1994.

[RP66] A. Rosenfeld and J. L. Pfalz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery*, 13:471–494, 1966.

[RP68] A. Rosenfeld and J. L. Pfalz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

[SM92] F. Y. -C. Shih and O. R. Mitchell. A mathematical morphology approach to Euclidean distance transformation. *IEEE Transactions on Image Processing*, 1:197–204, 1992.

[Vin92] L. Vincent. Morphological algorithms. In E. R. Dougherty, editor, *Mathematical Morphology in Image Processing*, chapter 8, pages 255–288. Marcel Dekker, New York, September 1992.

DT	*Parallel*	*Raster*	*Propagation*	b_4	b_8	b_O	b_{3-4}	b_{5-7-11}	b_E
[RP66]		X		X					
[RP68]	X								
[Bor86]	X	X		X	X	X	X	X	X
[SM92]	X			X	X		X	X	X
[Vin92]	X	X	X						
[HM94]	X								X

Table 1: Classification of DT algorithms and their structuring function decompositions.