Published in Image Processing On Line on 2014–09–01.
Submitted on 2013–02–12, accepted on 2013–04–19.
ISSN 2105–1232 © 2014 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2014.68

# A Streaming Distance Transform Algorithm for Neighborhood-Sequence Distances

Nicolas Normand[1], Robin Strand[2], Pierre Evenou[1], Aurore Arlicot[1]

[1] LUNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597, France
Nicolas.Normand@polytech.univ-nantes.fr
Pierre.Evenou@polytech.univ-nantes.fr
Aurore.Arlicot@polytech.univ-nantes.fr
[2] Centre for Image Analysis, Uppsala University, Sweden
http://www.cb.uu.se/~robin
robin@cb.uu.se

*Communicated by* Bertrand Kerautret *Demo edited by* Bertrand Kerautret

## Abstract

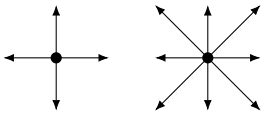
We describe an algorithm that computes a "translated" 2D Neighborhood-Sequence Distance Transform (DT) using a look up table approach. It requires a single raster scan of the input image and produces one line of output for every line of input. The neighborhood sequence is specified either by providing one period of some integer periodic sequence or by providing the rate of appearance of neighborhoods. The full algorithm optionally derives the regular (centered) DT from the "translated" DT, providing the result image on the fly, with a minimal delay, before the input image is fully processed. Its efficiency can benefit all applications that use neighborhood-sequence distances, particularly when pipelined processing architectures are involved, or when the size of objects in the source image is limited.

## Source Code

A C++ implementation of this algorithm and the on-line demo are accessible at the IPOL web page of this article[1].

**Keywords:** discrete distance; neighborhood-sequence distance; distance transform; Lambek-Moser inverse

[1] https://doi.org/10.5201/ipol.2014.68



(a) (b)

Figure 1: Neighborhoods of the NS-distance transforms. (a), and (b) are respectively the type 1 and 2 neighborhoods, $\mathcal{N}_1$ and $\mathcal{N}_2$.

# 1 Overview

Among discrete distances defined between points of $\mathbb{Z}^2$ many of them are path-based distances: the distance between two points is the length of the shortest path that connects the points. A path $\mathcal{P}$ from $p$ to $q$ is a sequence of points $\{p_0 = p, p_1, \ldots, p_n = q\}$ where each displacement between successive points, $\overrightarrow{p_{i-1}p_i}$, belongs to a set of neighbors $\mathcal{N}$ and its length is the number of displacements, $n$. Simple distances use a single type of neighborhood, namely the *1-neighborhood* $\mathcal{N}_1 = \{(0,0)\,;(\pm1,0)\,;(0,\pm1)\}$ (see Figure 1(a)) or *2-neighborhood* $\mathcal{N}_2 = \{(0,0)\,;(\pm1,0)\,;(0,\pm1)\,;(\pm1,\pm1)\}$ (see Figure 1(b)). These two distances are known as city-block, or $d_4$, and chessboard, or $d_8$. $d_4$ and $d_8$ are highly rotation dependent and have been generalized to combine both (or more) neighborhoods, either simultaneously using different weights (*weighted*, or *chamfer*, *distances*), either by changing the neighborhood depending on the travelled distance (*neighborhood-sequence*, or *NS*, *distances*). For NS-distances, used in the following, the neighborhood used at a given step of the path is driven by $B$, a sequence of 1 and 2: $\overrightarrow{p_{i-1}p_i} \in \mathcal{N}_{B(i)}$. For example, the octagonal distance uses alternatively the 1-neighborhood and the 2-neighborhood, corresponding to the 2-periodic sequence $B = (1, 2, 1, 2, \ldots)$. The city-block and chessboard distances are special cases of NS-distances with sequences $B = (1, \ldots)$ and $B = (2, \ldots)$, respectively. Several authors have proposed optimized sequences in order to minimize the rotational dependency of these distances. Due to the limited number of allowed steps and weights in the paths, a $3 \times 3$ neighborhood around each point is enough to extract reversible representations of objects, that can be used for, e.g., skeletonization, object decomposition, and resolution pyramids. For details, see [4] and the references therein.

Path-based distances naturally lead to propagation-based algorithms where the value at each pixel is derived from the values of its neighbors. In order to propagate the distance information in all directions, sequential algorithms need several scans of the image (typically two reversed-scans for chamfer distances, three different-order scans for NS-distances). Due to these different scans with different orders, DT algorithms need to keep a full image in memory.

The first algorithm proposed here computes an asymmetric generalized distance transform, with translated disks, using a single scan of the image. Input image pixels are processed on the fly and then discarded, so the needed memory is very low (about three lines of the image). The neighborhood sequence can be specified either by the rate of 2-neighbors or by a list of $n$ neighborhoods for $n$-periodic sequences.

A second algorithm is given to recover the centered, NS-distance transform. Its memory needs are a little higher because it has to relocate the disk centers according to their radii. However the general data flow is in the same raster scan order as the translated distance transform so both algorithms can be chained and the result pixels are given with minimal delay, before the input image is fully processed. Their efficiency can benefit all applications with discrete distances, particularly when pipelined processing architectures are involved, when the size of objects in the source image is limited or when the size is very large (or even infinite) in one direction.

## 1.1 Notations

The *distance transform* ($\mathrm{DT}_X$) of the binary image $X$ maps each pixel to its distance from the background (white) pixels:

$$\begin{aligned} \mathrm{DT}_X : \mathbb{Z}^2 &\to \mathbb{N} \\ p &\mapsto \min\left\{d(q,p) \,:\, q \in \mathbb{Z}^2 \setminus X\right\} . \end{aligned} \tag{1}$$

Figure 3(b) displays the distance transform of a simple image for the octagonal distance.

Alternatively, $\mathrm{DT}_X(p)$ can be interpreted in terms of the radius of largest disks included in $X$:

$$\mathrm{DT}_X(p) = \max\left\{r \,:\, \check{D}(p, r-1) \subset X\right\} , \tag{2}$$

or, equivalently:

$$\mathrm{DT}_X(p) \geq r \iff \check{D}(p, r-1) \subset X . \tag{3}$$

where $\check{D}(p,r)$ is the reflection of the disk $D(p,r)$ through its center $p$. $D(p,r)$ (resp. $\check{D}(p,r)$) is the set of points at a distance from (resp. to) $p$ not greater than $r$:

$$\begin{aligned} D(p,r) &= \left\{q \,:\, d(p,q) \leq r\right\} \\ \check{D}(p,r) &= \left\{q \,:\, d(q,p) \leq r\right\} . \end{aligned}$$

If $d$ is a *distance* (hence symmetric), $D(p,r)$ and $\check{D}(p,r)$ are equal. However, the distinction is important for *asymmetric distances* used in the following. Disks of NS-distances, including the simple distances $d_4$ and $d_8$, are iteratively produced by Minkowski sums:

$$D(p,r) = \begin{cases} \{p\} & r = 0 \\ D(p, r-1) \oplus N_B(r) & r > 0 \end{cases} . \tag{4}$$

Reflected disks $\check{D}(p,r)$ are similarly produced with the reflections of neighborhoods through the origin, $\check{N}_B(r)$. We recall that the Minkowski sum of two sets is the set of sums of pairs of elements, one from each set:

$$X \oplus Y = \left\{x + y \,:\, x \in X, y \in Y\right\} .$$

In the following, we denote by $\mathrm{DT}_X$ the distance transform of the set of points (or image) $X$ and $\mathrm{DT}'_X$ its asymmetric counterpart. In the same way, $\mathcal{N}$ and $\mathcal{N}'$ are the centered and translated neighborhoods, $d$ and $d'$, the regular distance and its asymmetric version. $\mathcal{N}'$ is equal to $\mathcal{N}$ up to a translation $\overrightarrow{t}$ such that the first point of $\mathcal{N}'$ in raster scan is the origin $O = (0,0)$. The reflected translated neighorhood $\check{\mathcal{N}}'$ is said to be in *forward scan condition* as well as the disks it generates. With $\mathcal{N}_1 = \left\{(0,0)\,;(\pm1,0)\,;(0,\pm1)\right\}$ and $\mathcal{N}_2 = \left\{(0,0)\,;(\pm1,0)\,;(0,\pm1)\,;(\pm1,\pm1)\right\}$, we have the translation vectors $\overrightarrow{t_1} = (0,1)$ and $\overrightarrow{t_2} = (1,1)$. The translated neighborhoods are then

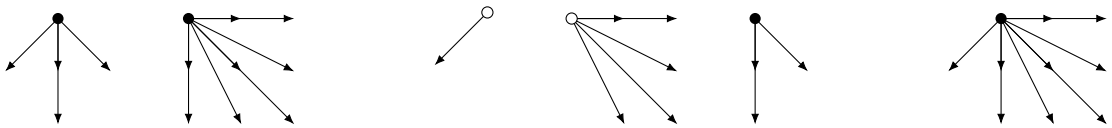
$$\mathcal{N}'_1 = \left\{(0,0)\,;(-1,1)\,;(0,1)\,;(1,1)\,;(0,2)\right\} \tag{5}$$

$$\mathcal{N}'_2 = \left\{(0,0)\,;(1,0)\,;(2,0)\,;(0,1)\,;(1,1)\,;(2,1)\,;(0,2)\,;(1,2)\,;(2,2)\right\} . \tag{6}$$

$\mathcal{N}'_1$ and $\mathcal{N}'_2$ are depicted in Figure 2(a) and 2(b).

It was shown that the following equation holds for every $\mathrm{DT}'$ (symmetric or not) and every point $p$ [3]:

$$\mathrm{DT}'_X(p) = \begin{cases} 0 & \text{if } p \notin X \\ \min\left\{\hat{C}_{\overrightarrow{v}}(\mathrm{DT}'_X(p - \overrightarrow{v})), \overrightarrow{v} \in \mathbb{Z}^2\right\} & \text{otherwise} \end{cases} . \tag{7}$$



(a) (b) (c) (d) (e) (f)

Figure 2: Neighborhoods used for the translated NS-distance transform. (a), and (b) are respectively the type 1 and 2 translated neighborhoods, $\mathcal{N}'_1$ and $\mathcal{N}'_2$. (c), (d) and (e) are respectively $\mathcal{N}'_1 \setminus \mathcal{N}'_2$, $\mathcal{N}'_2 \setminus \mathcal{N}'_1$ and $\mathcal{N}'_1 \cap \mathcal{N}'_2$, each set associated to a different sequence of displacement costs. (f) is the whole set of neighbors, $\mathcal{N}'_1 \cup \mathcal{N}'_2$, used for the translated NS-DT.

Here, $\hat{C}_{\overrightarrow{v}}(r) = \hat{c}_{\overrightarrow{v}}(r) + r$ and $\hat{c}_{\overrightarrow{v}}(r)$ represents the cost of the displacement in direction $\overrightarrow{v}$ when the distance already travelled is $r$ (with the assumption that $\hat{C}_{\overrightarrow{v}}$ never decreases [3]). In other words, $\hat{C}_{\overrightarrow{v}}(r)$ is the total length of a path formed by adding the displacement $\overrightarrow{v}$ to a path of length $r$. For Minkowski-based distances, $\hat{C}_{\overrightarrow{v}}(r)$ is simply the least distance value $s$ greater than $r$ where the displacement $\overrightarrow{v}$ is included in the neighborhood:

$$\hat{C}_{\overrightarrow{v}} = \min\left\{s \,:\, s > r, \overrightarrow{v} \in \mathcal{N}_{B(s)}\right\} .$$

The pseudo-distance presented here is strongly linked to the properties of non-decreasing integer sequences studied by Lambek and Moser and the method intensively uses their properties to iterate through sequence values.

The theoretical foundations of the distance used here was published in [2]. A version of these algorithms were the neighborhoods are further decomposed was described in [3].

# 2 Algorithm

## 2.1 Lambek-Moser Inverse of an Integer Sequence

Let the function $f$ define a non-decreasing sequence of integers $(f(1), f(2), \dots)$. The inverse sequence of $f$, denoted by $f^\dagger$, is a non-decreasing sequence of integers defined by [1]:

$$f(m) < n \iff f^\dagger(n) \not< m\,. \tag{8}$$

$f^\dagger(n)$ can be viewed as the count of values in the sequence $f$ that are less than $n$, or the last index of a value less than $n$ in $f$:

$$f^\dagger(n) = \max\left\{m \,:\, f(m) < n\right\} \text{ or } 0 \text{ if } f(1) \geq n \text{ or } \infty \text{ if } \forall m, f(m) < n\ .$$

$f^\dagger(f(m)+1)$ is then the last index of a value less than or equal to $f(m)$ and $f^\dagger(f(m)+1)+1$ the first index of a value greater than $f(m)$, i.e., the first index after $m$ where $f$ increases:

$$f^\dagger(f(m)+1)+1 = n \iff \forall j < n, f(j) \leq f(m) \text{ and } f(n) > f(m)\ . \tag{9}$$

## 2.2 Translated Distance Transform

The algorithm described in this section computes the translated distance transform of the set of points $X$.

$$\begin{aligned} \mathrm{DT}'_X(p) &= \min\left\{d'(q,p) \,:\, q \in \mathbb{Z}^2 \setminus X\right\} \\ &= \max\left\{r \,:\, \check{D}'(p, r-1) \subset X\right\}\ , \end{aligned} \tag{10}$$

where disks $D'(p,r)$ are translated versions of $D(p,r)$ in forward scan condition, $D'(p+\overrightarrow{t(r)},r) = D(p,r)$, for some radius-dependent translation vector $\overrightarrow{t(r)}$.

$$\mathrm{DT}'_X(p) \geq r \iff \check{D}'(p,r-1) \subset X \ . \tag{11}$$

Given a sequence of neighborhoods $B, \forall i > 0, B(i) \in \{1;2\}$, we define $\mathbf{j}_B(i)$, $j \in \{1;2\}$, as the number of times the value $j$ occurs in $B(1)\dots B(i)$. Since the distance disks are Minkowski-based, $\hat{C}_{\overrightarrow{v}}(r)$ is the index of the first appearance of $\overrightarrow{v}$ after $r$. If $\overrightarrow{v}$ only belongs to neighborhood $j$, it is clear that this index corresponds to the first index where $\mathbf{j}_B(i)$ increases after $r$ and $\hat{C}_{\overrightarrow{v}}(r)$ is then $\mathbf{j}_B^\dagger(\mathbf{j}_B(r)+1)+1$, according to equation (9). However, if $\overrightarrow{v}$ belongs to both neighborhoods, then its next appearance after $r$ is always $r+1$. We compute the cost of displacement in direction $\overrightarrow{v}$ when the distance $r$ has already been travelled as:

$$\hat{C}_{\overrightarrow{v}}(r) = \begin{cases} \hat{C}^1_{\overrightarrow{v}}(r) = \mathbf{1}_B^\dagger(\mathbf{1}_B(r)+1)+1 & \text{if } \overrightarrow{v} \in \mathcal{N}'_1 \text{ and } \overrightarrow{v} \notin \mathcal{N}'_2 \\ \hat{C}^2_{\overrightarrow{v}}(r) = \mathbf{2}_B^\dagger(\mathbf{2}_B(r)+1)+1 & \text{if } \overrightarrow{v} \notin \mathcal{N}'_1 \text{ and } \overrightarrow{v} \in \mathcal{N}'_2 \\ \hat{C}^{12}_{\overrightarrow{v}}(r) = r+1 & \text{if } \overrightarrow{v} \in \mathcal{N}'_1 \text{ and } \overrightarrow{v} \in \mathcal{N}'_2 \\ \infty & \text{otherwise} \end{cases},$$

where $f^\dagger$ denotes the Lambek-Moser inverse of the non-decreasing integer sequence $f$. When $B$ is a $l$-periodic sequence, so is $\hat{c}_{\overrightarrow{v}}(r) - r = \hat{C}_{\overrightarrow{v}}(r) - r$. The inverse Lambek-Moser sequences need only to be computed over a period and the result is an array of values of $\hat{c}_{\overrightarrow{v}}(r)$. The cost of this pre-computation is $\mathcal{O}(l)$, linear with the period of the sequence $l$, then $\hat{C}_{\overrightarrow{v}}(r)$ is computed in $\mathcal{O}(1)$ by looking up in a table of values of $\hat{c}_{\overrightarrow{v}}(r)$ and adding $r$. When $B$ is given by the occurrence rate $\tau$ of the value 2: $B(r) = \lfloor \tau r \rfloor - \lfloor \tau(r-1) \rfloor$, then $\hat{C}_{\overrightarrow{v}}(r)$ is computed on the fly, in $\mathcal{O}(1)$, without precomputation. Here, $\lfloor \bullet \rfloor$ denotes the floor function.

Algorithm 1 propagates distance transform values with a single raster scan of the image.

**Algorithm 1:** Generalized asymmetric distance transform

**input** : $X$, a set of points
**input** : $\mathcal{N}'$, neighborhood in forward scan condition
**input** : $\hat{C}_{\overrightarrow{v}}$, minimal absolute displacement costs
**ouput**: $\mathrm{DT}'_X$, generalized distance transform of $X$

```
foreach p in DT domain, in raster scan do
    if p ∉ X then
        DT'_X(p) ← 0
    end
    else
        l ← ∞
        foreach v in N' do
            l ← min{l ; Ĉ_v(DT'_X(p − v))}
        end
        DT'_X(p) ← l
    end
end
```

## 2.3 Centering the Distance Transform

The two neighborhoods are translated by vectors $\overrightarrow{t_1} = (0,1)$ and $\overrightarrow{t_2} = (1,1)$. The disk of radius $r$, produced by the sequence $B$, is translated by $(0,1)$ as many times 1 appears in the $r-1$ first terms

of $B$, i.e. $\mathbf{1}_B(r-1)$ and by $(1,1)$ as many times 2 appears in the $r-1$ first terms of $B$, i.e. $\mathbf{2}_B(r-1)$. Thus, $\overrightarrow{t(r)} = \mathbf{1}_B(r)\overrightarrow{t_1} + \mathbf{2}_B(r)\overrightarrow{t_2} = (\mathbf{2}_B(r), \mathbf{1}_B(r) + \mathbf{2}_B(r)) = (\mathbf{2}_B(r), r)$.

A link between the translated and the centered distance maps derives from equations (3) and (11):

$$\begin{aligned} \mathrm{DT}_X(p) \geq r &\iff D(p, r-1) \subseteq X \\ &\iff D'(p + \overrightarrow{t(r-1)}, r-1) \subseteq X \\ &\iff \mathrm{DT}'_X(p + \overrightarrow{t(r-1)}) \geq r \ . \end{aligned} \tag{12}$$

Consequently:

$$\mathrm{DT}_X(p) = r \iff \mathrm{DT}'_X(p + \overrightarrow{t(r)}) \leq r \leq \mathrm{DT}'_X(p + \overrightarrow{t(r-1)}) \ . \tag{13}$$

Centering the distance transform for pixel $p$ involves reading the values of $\mathrm{DT}'_X(p + \overrightarrow{t(r)})$ and $\mathrm{DT}'_X(p + \overrightarrow{t(r-1)})$, i.e., a pair of pixels, in the asymmetric distance transform, separated by $\overrightarrow{t(r)} - \overrightarrow{t(r-1)} = \overrightarrow{t_{B(r)}}$. In order to recenter the whole distance transform, all pairs of pixels $(p, p + \overrightarrow{t_1})$ and $(p, p + \overrightarrow{t_2})$ have to be scanned. In this phase, no further information propagation is involved so the order in which these pairs are considered is irrelevant. Algorithm 2 takes the result of the translated distance transform $\mathrm{DT}'$ and produces the centered distance transform DT. In the code archive, Algorithm 2 is implemented with a raster scan. It keeps a few rows of DT in memory while they are progressively completed by the algorithm (lines 9 to 11). The exact count of needed rows of DT depends on the maximal value of $\mathrm{DT}'$ on the currently processed row (due to the test in line 9). The algorithm visits exactly once each pixel in the input image $\mathrm{DT}'_X$ as well as each pixel in the result image $\mathrm{DT}_X$.

# 3 Implementation

The current implementation in C++ language works as a row image filter. It reads the input image either in pbm or png format from its standard input and writes the result to its standard output in pgm or png format. When only Algorithm 1 is used, a row of the translated distance transform is output each time a row of the input image is given. When combined with Algorithm 2, each row of the centered distance transform is output as soon as all its pixel values are valid, depending on the current maximal radius encountered in the input image.

## 3.1 Requirements

`LUTBasedNSDistanceTransform` currently uses netpbm and libpng for image I/O. Both libraries provide functions able to read images one row at a time. All source files are included in the archive accessible at the IPOL web page of this article[2].

## 3.2 Usage

```
LUTBasedNSDistanceTransform [-f filename] [-c] [-t (pgm|png)] \
                            (-4|-8|-r <num/den>|-s <sequence>)
```

### 3.2.1 Options

[2] https://doi.org/10.5201/ipol.2014.68

**Algorithm 2:** Computation of the centered distance transform DT from the generalized asymmetric distance transform DT′ produced by 1. In lines 8 and 11, $\mathbf{j}_B$ refers either to $\mathbf{1}_B$ or $\mathbf{2}_B$ depending on the value of $j$.

**input** : $\mathrm{DT}'_X$, translated distance map of $X$
**ouput**: $\mathrm{DT}_X$, centered distance map of $X$

1 **foreach** $p$ *in* $\mathrm{DT}'$ *domain* **do**
2  **if** $\mathrm{DT}'_X(p) = 0$ **then**
3  $\mathrm{DT}_X(p) \leftarrow 0$
4  **end**
5  **else**
6  **foreach** $j \in \{1;2\}$ **do**
7  $r \leftarrow \max\{1; \mathrm{DT}'(p + \overrightarrow{t_j})\}$
   // Least $r$ such that $r \geq \mathrm{DT}'(p - \overrightarrow{t_j})$ and $B(r) = j$
8  $r \leftarrow \mathbf{j}_B^{\dagger}(\mathbf{j}_B(r)) + 1$
9  **while** $r \leq \mathrm{DT}'(p)$ **do**
10  $\mathrm{DT}_X\big(p - \overrightarrow{t(r-1)}\big) \leftarrow r$
   // Next $r$ such that $B(r) = j$
11  $r \leftarrow \mathbf{j}_B^{\dagger}(\mathbf{j}_B(r) + 1) + 1$
12  **end**
13  **end**
14  **end**
15 **end**

```
-4              Use the city block distance.
-8              Use the chessboard distance.
-s sequence     One period of the sequence of neighborhoods given as a list of 1
                and 2 separated by " " or ",". Space characters must be escaped
                from the shell.
-r num/den      Ratio of neighborhood 2 given as the rational number num/den
                (with den >= num >= 0 and den > 0).
-c              Center the distance transform (the default is an asymmetric
                distance transform).
-f filename     Read from file "filename" instead of stdin.
-l              Flush output after each produced row.
-t format       Select output image format (pgm or png).
```

### 3.2.2 Examples

Regular octagonal distance transform (those commands produce the image displayed in Figure 3(b)):

```
"./LUTBasedNSDistanceTransform -r 1/2 -c < image.pbm" or
"./LUTBasedNSDistanceTransform -s '1 2' -c < image.pbm"
```

Translated octagonal distance transform (those commands produce the image displayed in Figure 3(a)):

```
"./LUTBasedNSDistanceTransform -r 1/2 < image.pbm" or
"./LUTBasedNSDistanceTransform -s '1 2' < image.pbm"
```

Manhattan distance transform ($\forall i, B(i) = 1$), also known as $d_4$, city-block:

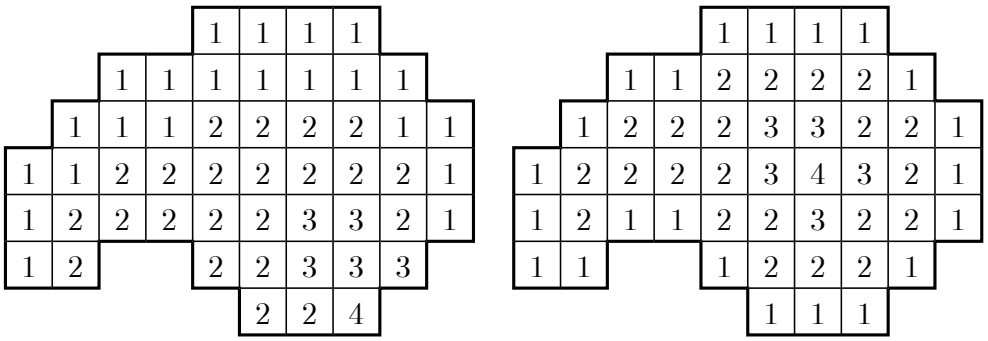
```
"./LUTBasedNSDistanceTransform -4 -c < image.pbm" or
"./LUTBasedNSDistanceTransform -r 0/1 -c < image.pbm" or
"./LUTBasedNSDistanceTransform -s '1' -c < image.pbm"
```

Chessboard distance transform ($\forall i, B(i) = 2$), also known as $d_8$:

```
"./LUTBasedNSDistanceTransform -8 -c < image.pbm" or
"./LUTBasedNSDistanceTransform -r 1/1 -c < image.pbm" or
"./LUTBasedNSDistanceTransform -s '2' -c < image.pbm"
```

# 4 Examples

The two following images show a translated distance transform as produced by Algorithm 1, and the centered distance transform produced by Algorithm 2.




(a) Translated (b) Centered

Figure 3: Result of distance transform algorithms for the octagonal distance (obtained with parameters `-r 1/2` or `-s '1 2'`).

# References

[1] J. Lambek and L. Moser, *Inverse and complementary sequences of natural numbers*, The American Mathematical Monthly, 61 (1954), pp. 454–458. http://dx.doi.org/10.2307/2308078.

[2] N. Normand, R. Strand, P. Evenou, and A. Arlicot, *Path-based distance with varying weights and neighborhood sequences*, in Discrete Geometry for Computer Imagery, vol. 6607 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 199–210. http://dx.doi.org/10.1007/978-3-642-19867-0_17.

[3] ——, *Minimal-delay distance transform for neighborhood-sequence distances in 2D and 3D*, Computer Vision and Image Understanding, 117 (2013), pp. 409–417. http://dx.doi.org/10.1016/j.cviu.2012.08.015.

[4] R. Strand, *Sparse object representations by digital distance functions*, in Discrete Geometry for Computer Imagery, vol. 6607 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 211–222. http://dx.doi.org/10.1007/978-3-642-19867-0_18.