A Weight Sequence Distance Function

Benedek Nagy[1,⋆], Robin Strand[2], and Nicolas Normand[3]

[1] Faculty of Informatics, University of Debrecen, Hungary
[2] Centre for Image Analysis, Uppsala University, Sweden
[3] LUNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597, Nantes, France
robin@cb.uu.se, nbenedek@inf.unideb.hu

Abstract. In this paper, a family of weighted neighborhood sequence distance functions defined on the square grid is presented. With this distance function, the allowed weight between any two adjacent pixels along a path is given by a weight sequence. We build on our previous results, where only two or three unique weights are considered, and present a framework that allows any number of weights. We show that the rotational dependency can be very low when as few as three or four unique weights are used. An algorithm for computing the distance transform (DT) that can be used for image processing applications is also presented.

1 Introduction

In a digital space (given for example by the pixels on a computer screen), not all properties of the Euclidean geometry are fulfilled. This is mainly due to the discrete (as opposed to continuous) structure of digital spaces. An example that shows that the Euclidean *distance function* has some disadvantages in digital spaces is the following: Circles, i.e., points of equal distance form a single point, are in general not connected in the usual sense: The eight points at distance $\sqrt{5}$ from a given point are disconnected with any of the common digital connectivities. Moreover, for most of the positive real values r there is not any point with this distance from any other point, see for example [6]. Also, computing with the Euclidean distance [11], in some cases, is very time consuming, especially for the so-called constrained distance transform, see the discussion in [14].

Therefore, the use of digital, i.e., path-based, distances is potentially very important for both digital image processing and computer graphics. For a path-based distance, each distance value attained is the cost of a connected path between two pixels in a square grid. In this aspect, the digital approach we follow is fundamentally different from the approach when Euclidean distances are computed. We believe that it is important to develop both the theory based on these digital distances and practical algorithms for image processing that effectively can utilize these distances.

The two digital distances first described in the literature are the city block and chessboard distances [12]. It is very easy to compute and use them, but they have very high rotational dependency (anisotropy). The theory of digital distances has developed rapidly

⋆ The work of Benedek Nagy is supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

C.L. Luengo Hendriks, G. Borgefors, R. Strand (Eds.): ISMM 2013, LNCS 7883, pp. 292–301, 2013.

from 1980's. The weighted (chamfer) distances [1], where the grid points together with costs to local neighbors form a graph in which the minimal cost path is the distance, is a well-known and often used concept in image processing. Contrary to weighted distances, the allowed steps may vary along the path with distances based on neighborhood sequences from a predefined set of steps [3]. For instance, by mixing the city block and chessboard neighborhood. In [18,16], the concept of weighted distances is generalized by allowing the size of the neighborhood to vary along the path. In this way, we get a distance function with potentially lower rotational dependency compared to when a fixed neighborhood is used. Recent results on Euclidean distance approximation are found in [4,2,7].

In this paper, we extend the idea presented in [15], where distances defined by three different local steps were considered. We allow using a fixed, but arbitrary large, number of possible local steps in the neighborhood sequence. Each of the allowed steps use only the 8-neighborhood of the pixels, but with different weights. Our main motivation is to provide a framework, to define digital distance functions that have as low rotational dependency as possible, that can be used to develop efficient image processing tools. Here, this is obtained by finding weight sequences that approximate the Euclidean distance.

Given a distance function, a *distance transform* is a transform where each element in a set is assigned the distance to the closest (as given by the distance function) element in a complementary set. The result of a distance transform is called a distance map. This tool is often used in image processing and computer graphics [5]. In digital geometry, the geometry of integer grids is used for building algorithms for, for example, image processing. It is very natural to define distance functions in this setting by minimal cost paths. In this paper, we present an algorithm for computing the distance map.

The structure of the paper is as follows. In the next section we present definitions and also some theoretical results, e.g., a formula to compute the point-to-point distance. In Section 3 parameter optimization is shown to obtain distances with low rotational dependency, i.e., approximating the Euclidean distance in this sense. Section 4 contains an algorithm for computing the distance map (DM) with some examples.

2 Theory

In this paper, we consider grid points with integer coordinates. Of course, in image processing, each grid point is associated with a picture element, pixel. In a city block (resp. chessboard distance), points with unit difference in at most one (resp. two) of the coordinates have unit distance. Here, we use the notion of 1- and 2-neighbors in the following sense: Two grid points $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in \mathbb{Z}^2$ are ρ-neighbors, $\rho \in \{1, 2\}$, if

$$\begin{aligned} |x_1 - x_2| + |y_1 - y_2| &\leq \rho \text{ and} \\ \max\{|x_1 - x_2|, |y_1 - y_2|\} &= 1. \end{aligned} \tag{1}$$

The points are *strict* ρ-neighbors if the equality in (1) is attained. Two points are *adjacent* if they are 2-neighbors. A neighborhood sequence (ns) B is a sequence

$B = (b(i))_{i=1}^{\infty}$ of neighborhood relations [3,8]. The shortest B-path between any two points can be computed by a greedy algorithm (see, e.g., [8]). A formula to compute B-distances can be found in [9,10] for the square grid. However, in this paper we use weight sequences instead of neighborhood sequences and will therefore use a modified description. A weight sequences is denoted $W = (w(i))_{i=1}^{\infty}$. A weight sequence W can be used as follows: In a W-distance, opposed to the B-distance, a weight to the 2-neighbors are always given by the weight sequence. The cost of a move to a 1-neighbor is 1 in every step, and the cost of a move to a strict 2-neighbor is given in the weight sequence.

In this work, we allow $m \in \mathbb{N},\ m \geq 2$ different neighborhood relations:

- a traditional 1-step is a step between 1-neighbors with unit weights, the sign ∞ denote these steps in W (practically, strict 2-steps are not allowed);
- a traditional 2-step is a step between 2-neighbors with unit weights, they are denoted by 1 in W;

and if $m > 2$, then let $\{w_3, \ldots, w_m\}$ be the used weight set and in these cases the further steps are:

- weighted 2-steps are steps between 1-neighbors with unit weights, and between strict 2-neighbors are steps with a weight w_k (where $3 \leq k \leq m$ and $1 \leq w_k \leq \infty$).

In this paper the weight sequence W can contain m weights of a predefined weight set, i.e., $w(i) \in \{1, \infty, w_3, \ldots, w_m\}$ for all $i \in \mathbb{N}, i > 0$.

A *path* in a lattice is a sequence of adjacent lattice points. A path $P_0, P_1, \ldots, P_n$ is a path of n steps where for all $i \in \{1, 2, \ldots, n\}$, P_{i-1} and P_i are adjacent.

The cost (weighted length) of a path is the sum of the weights along the path, i.e.,

$$\sum_{i=1}^{n} \delta_i, \text{ where } \delta_i = \begin{cases} w(i), & \text{if } P_{i-1} \text{ and } P_i \text{ are strict} \\ & \text{2-neigbors;} \\ 1, & \text{otherwise.} \end{cases}$$

When the weight sequence W is fixed we use the term W-path for paths having finite cost as defined by the weight sequence W. A W-path between P and Q is a *minimal cost* W-path if no other W-path between the points has lower cost. (If a step with weight ∞ has been taken, then the length of this path is ∞ and it is greater than any finite number.) The W-distance between P and Q is the cost of a minimal cost W-path between P and Q.

Regarding only the W-distances and paths with minimal costs they are obtained without any step with a weight value $w_i > 2$. This fact allows reducing our notation, the steps and so the values in the weight sequence W with weight ∞ (together with all values that are larger than 2) can be replaced by the same number (and it could be any number that is larger than 2). We use the notation ∞ for these steps in this paper. Based on this we can say that in our paths only weights between 1 and 2 play important role.

Example 1. Let the weight sequence $W = (1, 1.9, 1.8, 1, 1.5, \ldots)$. Then the shortest W-path from $(0, 0)$ to $(2, 2)$ includes two diagonal steps with weights $1 + 1.9 = 2.9$.

However the shortest W-paths from $(0,0)$ to $(3,3)$ is not a continuation of the former path, but consists of a diagonal step to $(1,1)$ with weight 1, then two consecutive 1-steps (to either $(1,2)$ or $(2,1)$ and, then, to $(2,2)$) and finally a diagonal step with weight 1: in this way the W-distance of $(0,0)$ and $(3,3)$ is 4. Comparing these shortest paths with four steps, one can reach the point $(3,3)$ in three steps from $(0,0)$, but the weight of these three diagonal steps 4.7 together.

The W-distance of $(0,0)$ and $(2,3)$ is 3.8 and it comes from the shortest path including a diagonal step (weight 1), then, we need a 1-step and diagonal step by weight 1.8.

By Example 1, one can see that greedy algorithm cannot be used to provide shortest paths. If a smaller weight appears after a larger weight in W, we may need this smaller in our shortest path, but it depends on both the weight sequence and on the difference of the coordinate values of the points.

2.1 Formula for Computing the Distance Function

Now we give a formula for computing the distance between any two grid points. The formula is used for finding optimal parameters in Section 3.

Let the weight sequence W, with the the weight set $\{1, \infty, w_3, \ldots, w_m\}$ and the point $(x,y) \in \mathbb{Z}^2$, where $x \geq y \geq 0$, be given. The number of steps in an optimal path from the point $\mathbf{0}(0,0)$ to the point (x,y) is between x and $x+y$ since the last case gives only 1-steps, which are always allowed. In this case, the distance is exactly $x+y$. The first case gives $x-y$ 1-steps and y 2-steps, so to get the path cost, we sum up the 1-steps and the y smallest weights of the first x elements. In the general case, we find the optimal value of $f = 0 \ldots y$ (that gives the number of 2-steps) by summing up the $x-y+2f$ 1-steps and the $y-f$ smallest weight among the first $x+f$ elements in the weight sequence (2-steps). The distance is defined for the f that gives the path with the lowest cost. This gives the formula

$$d(\mathbf{0},(x,y);W) = \min_{f=0..y} \left\{ x - y + 2f + \sum_{i \in I} w(i) \right\} \tag{2}$$

where the index set I contains the index of the smallest $y-f$ weight values among the first $x+f$ values of the weight sequence W. Since the roles of the x- and y-coordinate are similar, and our distance function is translation invariant, one can easily compute the W-distance of any pair of points of $\mathbb{Z}^2$ by our formula.

Our general approach consists several special cases:

- $W = (w_3)_1^\infty$ – traditional chamfer distance
- W contains only 1's and ∞'s – traditional distances based on neighborhood sequences ([3,8,9])
- W contains only 1's and w_3 – distances defined by weighted neighborhood sequences ([14])
- W contains only values 1, ∞, w_3 – distances defined by three types of local steps [15]

Note that, as opposed to the above mentioned first three usual cases, in our general case the greedy algorithm do not produce the optimal path and so the distance cannot be obtained by their help.

3 Parameter Optimization

In this section, we give some results on the approximation of the Euclidean distance. We find weight sequences that give a small difference between the Euclidean distance $d_E(\cdot,\cdot)$ and the weight sequence distance $d(\cdot,\cdot;W)$ between the point $\mathbf{0}$ and the point (x,y), where $x \geq y \geq 0$. See also [4]. The general case follows by symmetry as we discussed at the previous section about the formula.

Lemma 1. *If the weight sequence W is non-decreasing and all elements in W are smaller than or equal to 2, the distance in (2) is given by*

$$d(\mathbf{0},(x,y);W) = x - y + \sum_{i=1}^{y} w(i)$$

Proof. Since the lowest weights are the first in the weight sequence, we have

$$\min_{f=0..y}\left\{x-y+2f+\sum_{i\in I} w(i)\right\} = \min_{f=0..y}\left\{x-y+2f+\sum_{i=1}^{y-f} w(i)\right\},$$

where the sum from $i=1$ to $i=0$ is 0. Since the weights are smaller than or equal to 2, the optimum is attained for $f=0$, so

$$\min_{f=0..y}\left\{x-y+2f+\sum_{i=1}^{y-f} w(i)\right\} = x-y+\sum_{i=1}^{y} w(i). \qquad \square$$

Proposition 1. *Given an integer $x > 0$, the Euclidean distance values from $(0,0)$ to the set $\{(x,y) \in Z^2, 0 \leq y \leq x\}$ is given without errors by the weight sequence $\left(1+\sqrt{x^2+i^2}-\sqrt{x^2+(i-1)^2}\right)_{i=1..x}$.*

Proof. All weights in the weight sequence are smaller than 2 and the sequence is increasing, so by Lemma 1

$$\begin{aligned} d\left(\mathbf{0},(x,y);W\right) &= x-y+\sum_{i=1}^{y} w(i) \\ &= x-y+\sum_{i=1}^{y}\left(1+\sqrt{x^2+i^2}-\sqrt{x^2+(i-1)^2}\right) \\ &= x-y+\sqrt{x^2+y^2}-(x-y) \\ &= d_E(\mathbf{0},(x,y)). \end{aligned}$$

$\square$

The following remark follows by the construction of the index set I in the previous section.

Remark 1. Proposition 1 holds for any permutation of the weights in the weight sequence.

Given the number of weights in the weight set, Proposition 1 and Remark 1 gives a set of weights that optimally approximates the Euclidean distance on the border of a square. Now, a reasonable order of the weights are computed in order to obtain the weight sequence. This is done by using a greedy algorithm, Algorithm 1.

Algorithm 1. Algorithm for computing suboptimal weight sequence from the sequence obtained by Proposition 1.

Input: A sequence of weights $1 \leq w(i) < 2$, $i = 1, .., k$, obtained by Proposition 1.
Output: A sequence w' with suboptimal order of the weights.
Let $\mathcal{K} = \{1, 2, \ldots, k\}$ and $w' = (\infty, \infty, \ldots)$;
foreach $i = 1..k$ **do**
  $j' = \arg\min_{j \in \mathcal{K}} \left(\sum_{l=1..i} \left| d_E(i,l) - d(i,l;w'') \right|, \text{where } w''(m) = \begin{cases} w'(m) \text{ for } m < i \\ w(j) \text{ for } m = i \\ \infty \text{ for } m > i \end{cases} \right)$;
  $w'(i) \leftarrow w(j')$;
  $\mathcal{K} \leftarrow \mathcal{K} \setminus \{j'\}$;

Given a square centered in $(0, 0)$ (a chessboard disk of radius k), for $x = 0..k$, the weight that minimizes the difference to the Euclidean distance in the next step is added to the weight sequence. The weight sets obtained by Proposition 1, for increasing x, are listed in Table 1.

Table 1. Optimal weight sequences (with rounded weights) obtained by Proposition 1. The first column shows the number of weights obtained by Proposition 1.

#	w(1)	w(2)	w(3)	w(4)	w(5)
1	1.4142				
2	1.2361	1.5924			
3	1.1623	1.4433	1.6371		
4	1.1231	1.3490	1.5279	1.6569	
5	1.0990	1.2861	1.4458	1.5722	1.6679

In the (greedy) Algorithm 1, the mean absolute difference between $d_E(\cdot, \cdot)$ and $d(\cdot, \cdot; W)$ is minimized in each step up to a radius 50. The sequences of weights obtained are listed in Table 2.

Table 2. Suboptimal weight sequences obtained by Algorithm 1 using the weights in Table 1. The sequences show the first 20 indices of the corresponding weight sequence in Table 1. The first column shows the number of weights in the sequence obtained by Proposition 1.

#	weight sequence
1	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
2	1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1
3	2, 2, 1, 3, 1, 3, 2, 1, 3, 2, 1, 3, 2, 1, 3, 1, 3, 2, 1, 3
4	2, 3, 2, 1, 4, 3, 2, 1, 4, 2, 3, 1, 4, 2, 3, 1, 4, 2, 3, 1
5	3, 2, 4, 2, 3, 1, 5, 3, 2, 4, 1, 5, 3, 2, 4, 1, 5, 3, 2, 4

4 Distance Transform (DT)

The DT is a mapping from the image domain, a subset of the grid, to the range of the distance function. In a DT, each object grid point is given the distance from the closest background grid point. A modified version of a wave-front propagation algorithm can be used.

Now the formal definition of image is given.

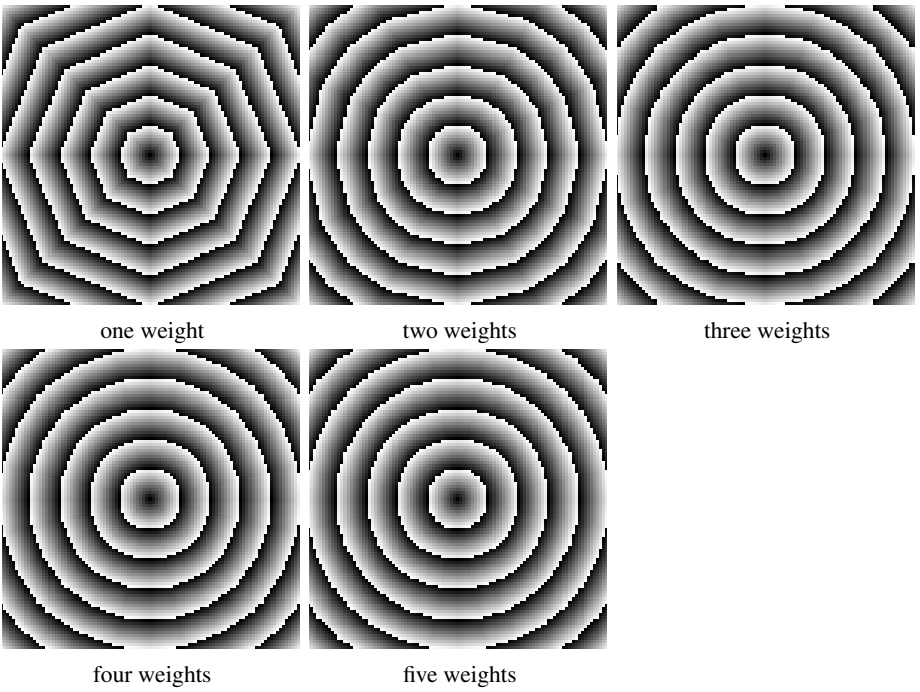

Fig. 1. Distance maps from a single point using the weights and sequences in Table 1 and Table 2. The distance values are shown modulo 10.

Definition 1. *The image domain is a finite subset of $\mathbb{Z}^2$ denoted by $\mathcal{I}$. We call the function $F : \mathcal{I} \longrightarrow \mathcal{R}_d$ an image, where $\mathcal{R}_d$ is the range of the distance function d.*

An *object* is a subset $X \subset \mathcal{I}$ and the *background* is $\overline{X} = \mathcal{I} \setminus X$. We assume that $X, \overline{X} \neq \emptyset$. We denote the distance map for path-based distances with DM_d, where the subscript d indicates what distance function is used.

Definition 2. *The* distance map DM_d *generated by the distance function $d(\cdot, \cdot; W)$ of an object $X \subset \mathcal{I}$ is the mapping*

$$DM_d : \mathcal{I} \to \mathcal{R}_d \textit{ defined by}$$

$$P \mapsto d\left(\overline{X}, P; W\right), \textit{ where}$$

$$d\left(\overline{X}, P; W\right) = \min_{Q \in \overline{X}} \left\{d\left(Q, P; W\right)\right\}.$$

In the case of W-distances with two weights, a minor modification of the Dijkstra algorithm (and with the same time complexity) can be used, see [17] and Theorem 4.1 in [13]. However, for multiple number of weights, this is not necessarily true. For W-distances, the used weights are also of importance, and they are determined by the *number of steps* of the minimal cost-path (not the cost). Therefore, in the general case of multiple weights presented here, we need to store this value also when propagating distance information. We define the auxiliary transform DM_s that holds the number of steps of the minimal cost path at each point, see Algorithm 2.

Note that we need to store not only the best distance values at the points, but the best values that are computed by various number of steps. Therefore for each point P a set $S(P)$ of pairs of values of the form (DM_d, DM_s) are stored with pairwise different DM_s. After the run of the algorithm for each point the minimal DM_d gives the result.

The novel Algorithm 2 shows how the distance map can be computed based on an extended optimal search (Dijkstra algorithm) and using the data structure described above. At the initialization the object points are the only points that are reached and it is done by 0 steps and with 0 cost. Then the border points of the object are listed in an increasing order by the minimal cost path already known for them. Actaully every point in the list has 0 cost, but the list will be updated by involving other points to where paths are already found. The **while** loop chooses (one of the) point(s) with minimal cost from the list since it is sure that we have the minimal cost path to this point already. Then in the loop the data of all neighbor points of the chosen point are updated by computing the cost of the new paths through the chosen point (having last step from the chosen point to the actual neighbor point). Therefore the algorithm holds the optimal distance attained at each point (as the usual algorithm), but this is done *for each* path length. So, if there are paths of different lengths ending up at the same point, distance information for each of the different path lengths are stored.

Algorithm 2. Computing DM for W-distances given by a weight sequence W.

Input: W and an object $X \subset \mathbb{Z}^2$.
Output: The distance map DM_d.
Initialization: Let $S(P) \leftarrow \{(0,0)\}$ for grid points $P \in \overline{X}$. Let $DM_d(P) = \min\{DM_d \mid (DM_d, DM_s) \in S(P)\}$. For all grid points $P \in X$ adjacent to $\overline{X}$: push $(P, DM_d(P))$ to the list L of ordered pairs sorted by increasing $DM_d(P)$.
while L *is not empty* **do**
  Pop $(P, DM_d(P))$ from L;
  foreach Q*:* Q, P *are strict* 2*-neighbors* **do**
    foreach *pair* $(DM_d, DM_s) \in S(P)$ **do**
      if $w(DM_s + 1) \leq 2$ **then**
        if *there is an element* $(DM'_d, DM_s + 1) \in S(Q)$ **then**
          if $DM'_d > DM_d + w(DM_s + 1)$ **then**
            Replace $(DM'_d, DM_s + 1)$ by $(DM_d + w(DM_s + 1), DM_s + 1)$ in $S(Q)$
          end
        end
        else
          Add $(DM_d + w(DM_s + 1), DM_s + 1)$ to $S(Q)$
        end
      end
    end
    Let $DM_d(Q) = \min\{DM'_d \mid (DM'_d, DM'_s) \in S(Q)\}$
    Push $(Q, DM_d(Q))$ to the ordered list L;
  end
  foreach Q*:* Q, P *are* 1*-neighbors* **do**
    foreach *pair* $(DM_d, DM_s) \in S(P)$ **do**
      if *there is an element* $(DM'_d, DM_s + 1) \in S(Q)$ **then**
        if $DM'_d > DM_d + 1$ **then**
          Replace $(DM'_d, DM_s + 1)$ by $(DM_d + 1, DM_s + 1)$ in $S(Q)$
        end
      end
      else
        Add $(DM_d + 1, DM_s + 1)$ to $S(Q)$
      end
    end
    Push $(Q, DM_d(Q))$ to L
  end
end

5 Conclusions

When the optimal parameters are used, the distance function we have presented has very low rotational dependency. With our method one uses digital (path-based) distance that approximate the Euclidean distance on the grid points with small error. Still, the distance is defined as the minimal cost-path and can therefore be used, for example, to compute the distance map in an efficient way. We believe that the proposed distance

function is potentially useful in many other image processing algorithms, for example for computing skeletons, or other algorithms where the low rotational independency is required.

References

1. Borgefors, G.: Distance transformations in digital images. Computer Vision, Graphics, and Image Processing 34, 344–371 (1986)
2. Celebi, M.E., Celiker, F., Kingravi, H.A.: On Euclidean norm approximations. Pattern Recognition 44(2), 278–283 (2011)
3. Das, P.P., Chakrabarti, P.P.: Distance functions in digital geometry. Information Sciences 42, 113–136 (1987)
4. Denev, A.: Digital distance functions defined by sequence of weights, Bachelor Thesis, Dept. of Information Technology, Uppsala University (2011)
5. Fabbri, R., da, F., Costa, L., Torelli, J.C., Bruno, O.M.: 2D Euclidean distance transform algorithms: A comparative survey. ACM Computing Surveys 40(1), 1–44 (2008)
6. Klette, R., Rosenfeld, A.: Digital geometry - geometric methods for digital picture analysis. Morgan Kaufmann (2004)
7. Mukherjee, J.: Hyperspheres of weighted distances in arbitrary dimension. Pattern Recognition Letters 34(2), 117–123 (2013)
8. Nagy, B.: Distance functions based on neighbourhood sequences. Publ. Math. Debrecen 63(3), 483–493 (2003)
9. Nagy, B.: Metric and non-metric distances on $\mathbb{Z}^n$ by generalized neighbourhood sequences. In: IEEE Proceedings of 4th International Symposium on Image and Signal Processing and Analysis (ISPA 2005), Zagreb, Croatia, pp. 215–220 (2005)
10. Nagy, B.: Distance with generalized neighbourhood sequences in nD and ∞D. Discrete Applied Mathematics 156(12), 2344–2351 (2008)
11. Ragnemalm, I.: The Euclidean distance transform in arbitrary dimensions. Pattern Recognition Letters 14(11), 883–888 (1993)
12. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. Journal of the ACM 13(4), 471–494 (1966)
13. Strand, R.: Distance Functions and Image Processing on Point-Lattices: with focus on the 3D face- and body-centered cubic grids. Ph.D. thesis, Uppsala University, Sweden (2008), `http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-9312`
14. Strand, R.: Weighted distances based on neighbourhood sequences. Pattern Recognition Letters 28(15), 2029–2036 (2007)
15. Strand, R., Nagy, B.: A weighted neighborhood sequence distance function with three local steps. In: IEEE Proceedings of 8th International Symposium on Image and Signal Processing and Analysis (ISPA 2011), Dubrovnik, Croatia, pp. 564–568 (2011)
16. Strand, R., Nagy, B., Borgefors, G.: Digital distance functions on three-dimensional grids. Theoretical Computer Science 412, 1350–1363 (2011)
17. Strand, R., Normand, N.: Distance transform computation for digital distance functions. Theoretical Computer Science 448, 80–93 (2012)
18. Yamashita, M., Ibaraki, T.: Distances defined by neighborhood sequences. Pattern Recognition 19(3), 237–246 (1986)