Shortest Route on Height Map Using Gray-Level Distance Transforms

Leena Ikonen and Pekka Toivanen

Lappeenranta University of Technology
P.O.Box 20, 53851 Lappeenranta, Finland
leena.ikonen@lut.fi

Abstract. This article presents an algorithm for finding and visualizing the shortest route between two points on a gray-level height map. The route is computed using gray-level distance transforms, which are variations of the Distance Transform on Curved Space (DTOCS). The basic Route DTOCS uses the chessboard kernel for calculating the distances between neighboring pixels, but variations, which take into account the larger distance between diagonal pixels, produce more accurate results, particularly for smooth and simple image surfaces. The route opimization algorithm is implemented using the Weighted Distance Transform on Curved Space (WDTOCS), which computes the piecewise Euclidean distance along the image surface, and the results are compared to the original Route DTOCS. The implementation of the algorithm is very simple, regardless of which distance definition is used.

1 Introduction

Finding the shortest path between two points on a three dimensional surface is a common optimization problem in many practical applications, e.g. robotic and terrain navigation, highway planning, and medical image analysis. By considering the digitized surface as a graph, variations of Dijkstra's classical path search algorithm become feasible (e.g. [4], [10]). A dynamic programming-based algorithm for computing distances of fuzzy digital objects is presented in [9].

This article presents an algorithm for finding optimal routes, or so called minimal geodesics, between two points on a gray-level height map. Other distance map approaches for path optimization include level sets propagation [3], and morphological grassfire algorithms [5]. Our algorithm is based on the Distance Transform on Curved Space (DTOCS presented in [13]), which calculates distances on a gray-level surface, when the gray-levels are understood as height values of the image surface. The Route DTOCS, first presented in [2], is developed further by using distance definitions, which give more accurate values for the global distances compared to the original chessboard distance transform. Particularly the piecewise Euclidean distance calculated with the Weighted Distance Transform on Curved Space (WDTOCS [13]) produces reliably optimal routes.

I. Nyström et al. (Eds.): DGCI 2003, LNCS 2886, pp. 308–316, 2003.

2 Definitions for Route DTOCS and WDTOCS

In the distance image produced by the DTOCS or the WDTOCS, every pixel in the calculation area X has a value which corresponds to the distance of that pixel to the nearest background pixel in X^C. The definition of the DTOCS for any calculation area X can be found in [13]. In the Route DTOCS the same distance metrics apply, but the complement area X^C is restricted to a single point, and the distance can be calculated according to the following, slightly simplified, definitions.

A discrete gray-level image is a function $\mathcal{G} : Z^2 \to N$, where N is the set of positive integers.

Definition 1. *Let $N_8(p)$ denote the set of all 8 neighbors of pixel p in Z^2. Pixels p and q are 8-connected if $q \in N_8(p)$. Let $N_4(p)$ denote the set of 4-connected neighbors, and $N_8(p) \setminus N_4(p)$ the set of diagonal neighbors. A discrete 8-path from pixel p to pixel s is a sequence of pixels $p = p_0, p_1, ..., p_n = s$, where every p_i is 8-connected to p_{i-1}, $i = 1, 2, ..., n$..*

Definition 2. *Let $\Psi(x, y)$ denote the set of all possible discrete 8-paths linking points $x \in X$ and $y \in X^C$. Let $\gamma \in \Psi(x, y)$ and let γ have n pixels. Let p_i and p_{i+1} be two adjacent pixels in path γ. Let $\mathcal{G}(p_i)$ denote the gray value of pixel p_i.*

The length of the path γ is defined by $\Lambda(\gamma) = \sum_{i=1}^{n-1} d(p_i, p_{i+1})$, where the definition of $d(p_i, p_{i+1})$, i.e. the distance between neighbor pixels p_i and p_{i+1} on the path, depends on the distance transform used. The Weigthed Distance Transform on Curved Space (WDTOCS) uses the Euclidean distance calculated with the Pythagoras' theorem from the height difference and the horizontal displacement of the two pixels:

$$d(p_i, p_{i+1}) = \begin{cases} \sqrt{|\mathcal{G}(p_i) - \mathcal{G}(p_{i+1})|^2 + 1}\ , & p_{i+1} \in N_4(p_i) \\ \sqrt{|\mathcal{G}(p_i) - \mathcal{G}(p_{i+1})|^2 + 2}\ , & p_{i+1} \in N_8(p_i) \setminus N_4(p_i) \end{cases} \tag{1}$$

In the chessboard DTOCS the distance is defined as the height (gray-level) difference between the pixels, plus one for the horizontal displacement:

$$d(p_i, p_{i+1}) = |\mathcal{G}(p_i) - \mathcal{G}(p_{i+1})| + 1 \tag{2}$$

The distance can also be defined using separate height and pixel-to-pixel displacements as in DTOCS, but using the accurate horizontal distance between diagonal neighbors:

$$d(p_i, p_{i+1}) = \begin{cases} |\mathcal{G}(p_i) - \mathcal{G}(p_{i+1})| + 1 & , \ p_{i+1} \in N_4(p_i) \\ |\mathcal{G}(p_i) - \mathcal{G}(p_{i+1})| + \sqrt{2}\ , & p_{i+1} \in N_8(p_i) \setminus N_4(p_i) \end{cases} \tag{3}$$

Definition 3. *The distance image $\mathcal{F}^*(x)$ when $X^C = \{y\}$ is*

$$\mathcal{F}^*(x) = \begin{cases} \min\limits_{\gamma \in \Psi}(\Lambda(\gamma))\ , & x \in X \\ 0 & , \ x \in X^C \end{cases} \tag{4}$$

The same distance image definition is used for the WDTOCS, the DTOCS and for the distance transform using $\sqrt{2}$ as the horizontal displacement between diagonal neighbors. The definition of neighbor-distances $d(p_i, p_{i+1})$ used in calculating the path length $\Lambda(\gamma)$ determines which version of the distance transform is produced by the algorithm.

3 The Distance Transformation Algorithms

The two-pass algorithm (see [13]) for calculating the DTOCS or the WDTOCS image $\mathcal{F}^*(x)$ is a sequential local operation (see [7]). The algorithm requires two images: the original gray-level image $\mathcal{G}(x)$ and a binary image $\mathcal{F}(x)$, which determines the region(s) in which the transformation is performed. The calculation area X in $\mathcal{F}(x)$ is initialized to max (the maximal representative number of memory) and the complement area X^C to 0.

The first computation pass proceeds using the mask $M_1 = \{p_{nw}, p_n, p_{ne}, p_w\}$ in figure 1 rowwise from the top left corner of the image, substituting the middle point $\mathcal{F}(p_c)$ with the distance value

$$\mathcal{F}_1^*(p_c) = \min[\mathcal{F}(p_c),\ \min_{p \in M_1}(\Delta(p) + \mathcal{F}_1^*(p))] \tag{5}$$

The distance $\Delta(p)$ between pixels p_c and p is calculated according to the definition of the distance transformation that is used:

$$\text{WDTOCS: } \Delta(p) = \begin{cases} \sqrt{|\mathcal{G}(p) - \mathcal{G}(p_c)|^2 + 1}\ , & p \in N_4(p_c) \\ \sqrt{|\mathcal{G}(p) - \mathcal{G}(p_c)|^2 + 2}\ , & p \in N_8(p_c) \setminus N_4(p_c) \end{cases} \tag{6}$$

$$\text{DTOCS: } \Delta(p) = |\mathcal{G}(p) - \mathcal{G}(p_c)| + 1 \tag{7}$$

$$\sqrt{2}\text{-DTOCS: } \Delta(p) = \begin{cases} |\mathcal{G}(p) - \mathcal{G}(p_c)| + 1\ , & p \in N_4(p_c) \\ |\mathcal{G}(p) - \mathcal{G}(p_c)| + \sqrt{2}\ , & p \in N_8(p_c) \setminus N_4(p_c) \end{cases} \tag{8}$$

The backward pass uses the mask $M_2 = \{p_e, p_{sw}, p_s, p_{se}\}$ in figure 1 replacing the distance value $\mathcal{F}_1^*(p_c)$ calculated by the forward pass with the new value

$$\mathcal{F}^*(p_c) = \min[\mathcal{F}_1^*(p_c),\ \min_{p \in M_2}(\Delta(p) + \mathcal{F}^*(p))] \tag{9}$$

If the original gray-level map is complex, the two calculation passes may have to be repeated several times to get the perfect distance map (see [11]). The distance image $\mathcal{F}^*(x)$ is used instead of the binary image $\mathcal{F}(x)$ for the next computation pass repeatedly until the DTOCS algorithm has converged to the globally optimal distances.

4 The Shortest Route Algorithm

The shortest route algorithm is based on calculating two distance maps, one for each endpoint of the desired route. Assuming we have a gray-level map $G(x)$

p_{nw}	p_n	p_{ne}
p_w	(p_c)	

	(p_c)	p_e
p_{sw}	p_s	p_{se}

Fig. 1. The masks for calculating the DTOCS. The left mask M_1 is used in the forward calculation pass, and the right mask M_2 in the backward pass.

and want to find an optimal route from point a with gray-level value (i.e. height) $G(a)$ to point b with value $G(b)$, we initialize the binary images $\mathcal{F}_a(x)$ and $\mathcal{F}_b(x)$ with $X_a^C = \{a\}$ and $X_b^C = \{b\}$ respectively. Using these two images, we calculate the distance images $\mathcal{F}_a^*(x)$ and $\mathcal{F}_b^*(x)$ with one of the distance transformation algorithms. In the resulting distance maps each value corresponds to the distance between point x and point a (or b respectively) along an 8-connected path that is optimal according to the distance definition of the used algorithm, WDTOCS, DTOCS or $\sqrt{2}$-DTOCS. It can be noted that $\mathcal{F}_a^*(b)$ as well as $\mathcal{F}_b^*(a)$ equals the length of the shortest route between points a and b, but the route itself can not be seen in the separate maps. Using the two maps we define the route distance:

$$\mathcal{D}_{\mathcal{R}}(x) = \mathcal{F}_a^*(x) + \mathcal{F}_b^*(x) \tag{10}$$

For each point x the value $\mathcal{D}_{\mathcal{R}}(x)$ is the length of the shortest path from point a to b that passes through point x. The value $\mathcal{F}_a^*(x)$ is the shortest distance from a to x, and $\mathcal{F}_b^*(x)$ is the shortest distance from x to b, and these optimal subpaths form an optimal path (see [6]). The equal distance propagation curves in [3] are combined similarly to form minimal geodesics. Now the optimal route from a to b is the set of points, for which the route distance is minimal. We define the route:

$$\mathcal{R}(a,b) = \{\, x \mid \mathcal{D}_{\mathcal{R}}(x) = \min_x \mathcal{D}_{\mathcal{R}}(x)\} \tag{11}$$

There can be several optimal paths, and the set $\mathcal{R}(a,b)$ contains all points that are on any optimal path, so this method does not provide an analytical description of a distinct route (e.g. a sequence of pixels). However, the routes can be visualized by marking the set of pixels $\mathcal{R}(a,b)$ on the original image. In WDTOCS and $\sqrt{2}$-DTOCS real values are used in the calculations, but the route distance $\mathcal{D}_{\mathcal{R}}(x)$ is rounded up to nearest integer before finding the points with the minimal distance. To summarize, **the shortest route algorithm** is:

1. Calculate the distance image $\mathcal{F}_a^*(x)$ from source point a
2. Calculate the distance image $\mathcal{F}_b^*(x)$ from destination point b
3. Calculate the route distance $\mathcal{D}_{\mathcal{R}}(x) = \mathcal{F}_a^*(x) + \mathcal{F}_b^*(x)$
4. Mark points with $\mathcal{D}_{\mathcal{R}}(x) = \min_x \mathcal{D}_{\mathcal{R}}(x)$ as points on optimal route $\mathcal{R}(a,b)$

5 Experiments and Results

This section demonstrates how the shortest route algorithm works, and compares the results of implementations with different distance definitions. Figure

2 presents a step by step application of the algorithm. Figure 2 a) is the original gray-level image. Figures 2 b) and 2 c) show the DTOCS-images $\mathcal{F}_a^*(x)$ and $\mathcal{F}_b^*(x)$ calculated from the endpoints a and b (marked with 'x'). As the distance function is symmetrical, it does not matter which endpoint corresponds to a and which to b. Figure 2 d) shows the route distance image, i.e. the sum of the DTOCS-images. Images 2 b)–d) are scaled to gray-levels, but original distance values, which can be beyond 255, are used in the calculation of $\mathcal{D}_{\mathcal{R}}(x)$. Figure 2 e) presents the final result, i.e. the points in set $\mathcal{R}(a, b)$. Figure 2 f) presents the same route calculated with the WDTOCS. It can be seen that for the complex image surface representing varying terrain the route is very similar, but sharper than the route by the DTOCS.

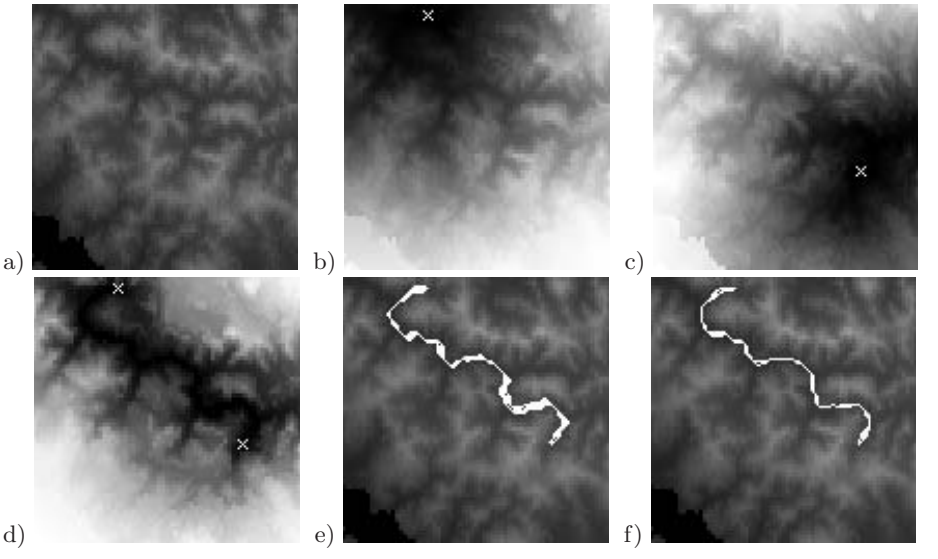

Fig. 2. a) Original image, b) distance from source point, c) distance from destination point, d) sum of distance images, e) route by DTOCS, f) route by WDTOCS.

A sample application, where the shortest route idea is used to solve a labyrinth, was presented in [2]. Figure 3 a) shows the route through a labyrinth produced by the original Route DTOCS. The algorithm needs a threshold segmented image, where labyrinth paths get value zero and walls get a very high value. Then the shortest path from the entrance to the exit of the labyrinth is the route through the labyrinth. It can be seen in figure 3 a) that the route makes seemingly extra 90° corners when calculated with the chessboard DTOCS.

The explanation to this problem is visualized in figure 4. The route from point A to B that passes through point x is just as short as as the intuitively optimal straight route, as there are as many pixel-to-pixel displacements on both routes. Consequently, there are several optimal discrete 8-connected paths through the labyrinth, and as the route is defined as the set of all points that are on any

optimal path, the visualized route becomes wide. Figure 3 b) shows how the route width decreases when the longer distance between diagonal pixels is taken into account according to equation 3.

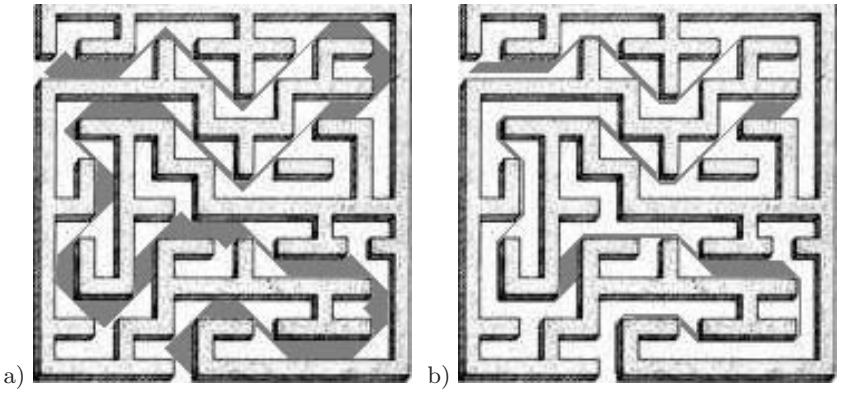

Fig. 3. a) Route through labyrinth by DTOCS b) Route through labyrinth by DTOCS with $\sqrt{2}$ diagonal distances.

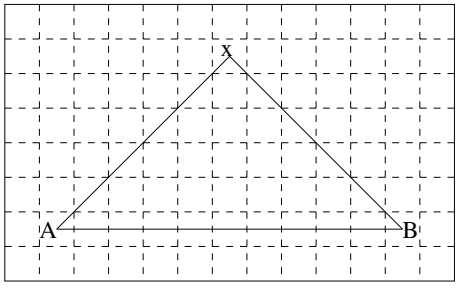

Fig. 4. Two of several possible routes from point A to B on a flat image surface according to the chessboard distance definition. The DTOCS distance is the same along the route through point x as along the straight line, as there are as many pixels on both routes (each square represents a pixel).

Tests with a gray-scale-ball image show similar results. The routes between the endpoints of the horizontal diameter of the half-sphere are too wide, when calculated with the basic Route DTOCS (figure 5 a), but introducing the $\sqrt{2}$-factor to the diagonal neighbor distances makes the routes as optimal as can be expected of discrete 8-connected paths (figure 5 b). Using the Euclidean neighbor distances of the WDTOCS changes the result dramatically, i.e. the algorithm finds the route across the half-sphere rather than around it (figure 5 c). The differing route lengths are partly a result of the digitization of the sphere function. Figure 6 shows a cross-section and a horizontal projection of a digital ball with few pixels. The digitization error is smaller but still present when

using a higher resolution ball image. Another big factor is that the variation in surface height increases the WDTOCS-distances less than the distances of the transforms that add the height difference to the horizontal displacement. The DTOCS-distance across the ball along the route the WDTOCS algorithm finds optimal (as in figure 5 c) would be clearly longer than the WDTOCS-distance, as each neighbor-distance $\sqrt{d^2+1}$ is replaced with $d+1$, where d is the height difference of the neighbor pixels.

When gray-level variations, i.e. height differences are large, the effect the horizontal displacements have on the distance value decreases in WDTOCS, whereas it stays constant in DTOCS and $\sqrt{2}$-DTOCS. The application determines which approach is better. If the transformation is used to approximate actual distances along a real surface, using the piecewise Euclidean distance of WDTOCS is justified. If the gray-level differences represent a different type of cost than the horizontal displacements, the transformations adding horizontal and vertical distances may work better and be more easily scalable. To modify the effect the height differences have on the distance transform, the original image can be scaled before applying the transformation. Alternatively, a weighting factor can be added to the height difference in equations 1, 2 and 3.

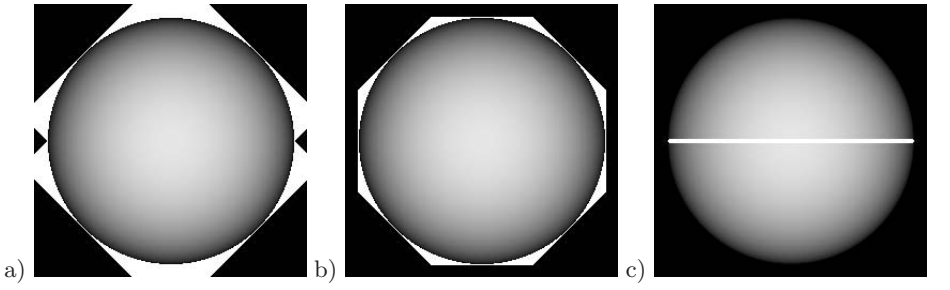

Fig. 5. a) Route by DTOCS, b) Route by DTOCS with $\sqrt{2}$ diagonal distance, c) Route by WDTOCS.

6 Discussion

In previous work, the DTOCS algorithm has mostly been used to calculate local distances. For example in image compression (see e.g. [12]) distance values are used to measure the variation of the image surface. More control points need to be stored from image areas, where local distances are high, i.e where gray-level values change rapidly. In such applications the chessboard distance transform works well enough, and the use of integer approximations of distance values is justified to save computation time and space. However, the route optimization algorithm computes global distances across the whole image, and the approximation error of the chessboard distance accumulates. Particularly on smooth and simple image surfaces, the chessboard Route DTOCS performs poorly, and using the WDTOCS produces more reliable optimal routes.

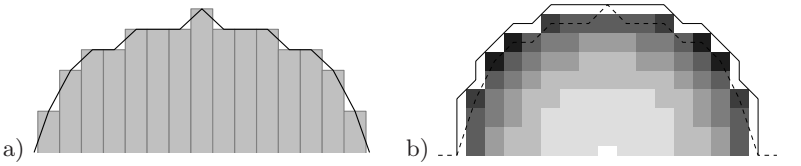

Fig. 6. a) Cross section of digititized ball with the WDTOCS route across the ball. The height of the bars corresponds to gray-level values. b) flat projection of digitized ball with the $\sqrt{2}$-DTOCS route around the ball, and the shape of the WDTOCS-route marked with dashed line for comparison. Each square represents a pixel.

The distance transform using $\sqrt{2}$ as the diagonal pixel-to-pixel displacement is an interesting hybrid of chessboard and Euclidean distance definitions, as the locally Euclidean distance is used as the horizontal pixel-to-pixel displacement, but the height difference is calculated just as in the chessboard DTOCS. The theoretical basis for this hybrid distance transform may not be as solid as for the DTOCS and the WDTOCS, but in route optimization it can give some interesting results. For example in the labyrinth application the horizontal displacements form the desired route, and the values of the gray-level differences are not significant, as long as distances along low paths are clearly shorter than distances over high walls. Other obstacle avoidance problems can be solved using the route optimization algorithm, and treating the horizontal and vertical displacements differently can be practical.

Using the piecewise Euclidean distances of WDTOCS gives the most accurate approximations for distances along the image surface. If the slightly heavier computation of floating point values instead of integers is not a problem, the WDTOCS algorithm should be used to get the best results in route optimization. A question for future research is whether we can define integer kernel distances, which approximate the Euclidean distance more accurately than the DTOCS. Borgefors [1] showed that using local distances 3 and 4 for square and diagonal neighbors in binary images actually gives a better approximation of Euclidean distance along the horizontal image plane than the distances 1 and $\sqrt{2}$ used here. Extending the ideas to gray-level images requires further investigation into how the height differences affect, and how they should affect the distance transformation.

References

1. Borgefors, G.: Distance Transformations in Digital Images. Computer vision, Graphics, and Image Processing, 34 (1986) 344–371
2. Ikonen, L., Toivanen, P., Tuominen, J.: Shortest Route on Gray-Level Map using Distance Transform on Curved Space. Proc. of Scandinavian Conference on Image Analysis (2003) 305–310
3. Kimmel, R., Amir, A. and Bruckstein A.: Finding Shortest Paths on Surfaces Using Level Sets Propagation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 17, no. 6 (1995) 635–640

4. Kimmel, R. and Kiryati, N.: Finding Shortest Paths on Surfaces by Fast Global Approximation and Precise Local Refinement. International Journal of Pattern Recognition and Artificial Intelligence, vol 10 (1996) 643–656
5. Lin P., Chang S.: A Shortest Path Algorithm for a Nonrotating Object Among Obstacles of Arbitrary Shapes. IEEE Transactions on Systems, Man, and Cybernetics, vol 23, no 3 (1993) 825–833
6. Piper J., Granum E.: Computing Distance Transformations in Convex and Non-Convex Domains. Pattern Recognition, vol 20, no 6 (1987) 599–615
7. Rosenfeld A., Pfaltz, J. L.: Sequential Operations in Digital Picture Processing. Journal of the Association for Computing Machinery, vol 13, no 4 (1966) 471–494
8. Rosin, P., West, G.: Salience Distance Transforms. Graphical Models and Image Processing, vol 56, no 6 (1995) 483–521
9. Saha P. K., Wehrli F. W., Gomberg B. R.: Fuzzy Distance Transform: Theory, Algorithms and Applications. Computer Vision and Image Understanding 86 (2002) 171–190
10. Saab, Y. and VanPutte M.: Shortest Path Planning on Topographical Maps. IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans, vol 29, no 1 (1999) 139–150
11. Toivanen, P. J.: Convergence properties of the Distance Transform on Curved Space (DTOCS). Proc. of Finnish Signal Processing Symposium (1995) 75–79
12. Toivanen, P. J.: Image Compression by Selecting Control Points Using Distance Function on Curved Space, Pattern Recognition Letters 14 (1993) 475–482
13. Toivanen, P.: New geodesic distance transforms for gray-scale images. Pattern Recognition Letters, 17 (1996) 437–450