Shortest Route on Gray-Level Map Using Distance Transform on Curved Space

Leena Ikonen, Pekka Toivanen, and Janne Tuominen

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

Abstract. The Distance Transform on Curved Space (DTOCS) can be used to calculate distances on a gray-level surface, but the route along which the shortest distance is found, is lost during the calculation. In this article a new method for finding and visualizing the shortest path between two points on a gray-level height map is presented. The method is simple to implement, and example route images show that it produces good results.

1 Introduction

Finding the shortest path, or the so called minimal geodesic, between two points on a three dimensional surface is an important optimization problem. Applications include robotic motion planning and navigation, terrain navigation, medical image analysis etc. By considering the digitized surface as a graph, numerous path search methods, like the classical Dijkstra's algorithm, best first search and A* search, become feasible. For example Saab and VanPutte present a geometrically modified Dijkstra search for path planning on a topographical map [6]. Kimmel and Kiryati use graph search to get an initial approximation of the shortest path and then refine it by curve shortening [3].

This article presents a new method for determining minimal length paths between two points on a gray-level map. The algorithm is based on the Distance Transform on Curved Space (DTOCS presented in [9]), which calculates distances on a gray-level surface, when the gray-levels are understood as height values of the image surface. Other distance map approaches for path optimization include level sets propagation presented in [2]. A grassfire algorithm by gray-scale morphology, as in [4], could also be used similarly.

2 Definitions for Route DTOCS

A discrete gray-level image is a function $\mathcal{G} : Z^2 \rightarrow N$, where N is the set of positive integers. Let $N_8(p)$ denote the set of 8 neighbors of pixel p in Z^2.

In the distance image produced by the DTOCS, every pixel in the calculation area X has a value which corresponds to the distance of that pixel from the nearest background pixel in X^C. The set X^C can also be defined as the set of feature pixels to compute distances to the nearest feature as in for example [8]. The definition of the DTOCS for any calculation area X can be found in [9]. 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.

Definition 1. *Let $N_8(p)$ denote the 8 neighbors of pixel p in Z^2. Pixels p and q are 8-connected if $q \in N_8(p)$. 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}.*

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.*

Definition 3. *Along a discrete 8-path the distance between neighbor pixels p_i and p_{i+1} is $d(p_i, p_{i+1}) = |\mathcal{G}(p_i) - \mathcal{G}(p_{i+1})| + 1$, and the length of the path γ is defined by $\Lambda(\gamma) = \sum_{i=1}^{n-1} d(p_i, p_{i+1})$.*

Definition 4. *The DTOCS distance image $\mathcal{F}^*(x)$ when $X^C = \{y\}$ is*

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

3 The DTOCS Algorithm

The sequential two-pass algorithm for calculating the DTOCS image $\mathcal{F}^*(x)$ presented in [9] 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 in direct video order, i.e. rowwise from the top left corner of the image, substituting the middle point $\mathcal{F}(p_c)$ with the distance value

$$\mathcal{F}_1^*(p_c) = \min[\mathcal{F}(p_c), \ \min_{p \in M_1}(\Delta(p) + \mathcal{F}_1^*(p))] \tag{1}$$

The distance between pixel p_c and its neighbor p is $\Delta(p) = |\mathcal{G}(p) - \mathcal{G}(p_c)| + 1$ according to definition 3.

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

$$\mathcal{F}^*(p_c) = \min[\mathcal{F}_1^*(p_c), \ \min_{p \in M_2}(\Delta(p) + \mathcal{F}^*(p))] \tag{2}$$

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 [7]). The DTOCS map $\mathcal{F}^*(x)$ is used as the "binary" image $\mathcal{F}(x)$ for the next computation pass repeatedly until the DTOCS algorithm has converged to the globally optimal distances.

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 DTOCS. The left mask M_1 is used in the forward calculation pass, and the right mask M_2 in the reverse pass.

4 The Shortest Route Algorithm

The method for finding the shortest route on a gray-level surface is based on calculating two DTOCS distance maps, one for each of the points between which we want to find the optimal route. Assuming we have a gray-level image $G(x)$ and want to find an optimal route from point a with gray-level value $G(a)$ 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, which in practice have the single pixel a (or b respectively) set to 0 and all other pixels to max, we individually calculate the two DTOCS images $\mathcal{F}_a^*(x)$ and $\mathcal{F}_b^*(x)$. In the resulting distance maps each value corresponds to the distance between point x and point a (or b respectively) along an optimal 8-connected path. 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. Using the two DTOCS images, we define the route distance:

$$\mathcal{D}_{\mathcal{R}}(x) = \mathcal{F}_a^*(x) + \mathcal{F}_b^*(x) \tag{3}$$

For each point x the value of $\mathcal{D}_{\mathcal{R}}(x)$ is the length of the shortest path from point a to b that passes through point x. This is evident, since $\mathcal{F}_a^*(x)$ is the shortest distance from a to x, and $\mathcal{F}_b^*(x)$ is the shortest distance from x to b. The proof about minimal subpaths in [5] can be modified to prove that the minimal subpaths from a to x and from x to b form a minimal path between a and b. The equal distance propagation curves in [2] are combined similarly to form minimal geodesics. So the route, i.e. the shortest path from a to b, is the set of points x, for which the route distance is minimal. We define the route:

$$\mathcal{R}(a,b) = \{\, x \mid \mathcal{D}_{\mathcal{R}}(x) = \min_x \mathcal{D}_{\mathcal{R}}(x)\} \tag{4}$$

There can be several optimal paths, but they are all of the same length. The set $\mathcal{R}(a,b)$ contains all points that are on any optimal path, so this method can 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 defined by equation (4) on the original image.

To summarize, **the algorithm** for finding the shortest path is:

1. Calculate the DTOCS image $\mathcal{F}_a^*(x)$ from source point a
2. Calculate the DTOCS 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

The figures in this section show some example routes. Figure 2 demonstrates how the algorithm proceeds. Image 2 a) is the original gray-level map image. Images 2 b) and 2 c) show the DTOCS-images $\mathcal{F}_a^*(x)$ and $\mathcal{F}_b^*(x)$ calculated from the end points a and b (marked with 'x'). The route distance function is symmetrical, i.e. it does not matter which end point corresponds to a and which to b. Image 2 d) shows the route distance image, i.e. the sum of the DTOCS-images. The images 2 b)–d) are scaled to standard gray-levels, but the original distance values, which can be beyond 255, are used in the calculation of $\mathcal{D}_{\mathcal{R}}(x)$. Image 2 e) presents the final result, i.e. the points in which the route distance $\mathcal{D}_{\mathcal{R}}(x)$ is minimal. Image 2 f) is another example route on the same image, i.e. the endpoints of the route are different than in the images 2 b)–e).

Figure 3 shows an experiment on a different kind of surface. Similar "eggbox" surfaces have been used in [2] and [3] to demonstrate the performance of shortest path algorithms.

Figure 4 shows a sample application, where the shortest route idea is used to solve a labyrinth. The labyrinth solver needs a labyrinth image, which can be segmented into paths and walls using a threshold. Paths get value zero and walls get a very large height value (2048 in the sample labyrinth of size 170x170 pixels) to make sure the distance transformation does not create illegal shortcuts over walls. Then the shortest path is the route through the labyrinth.

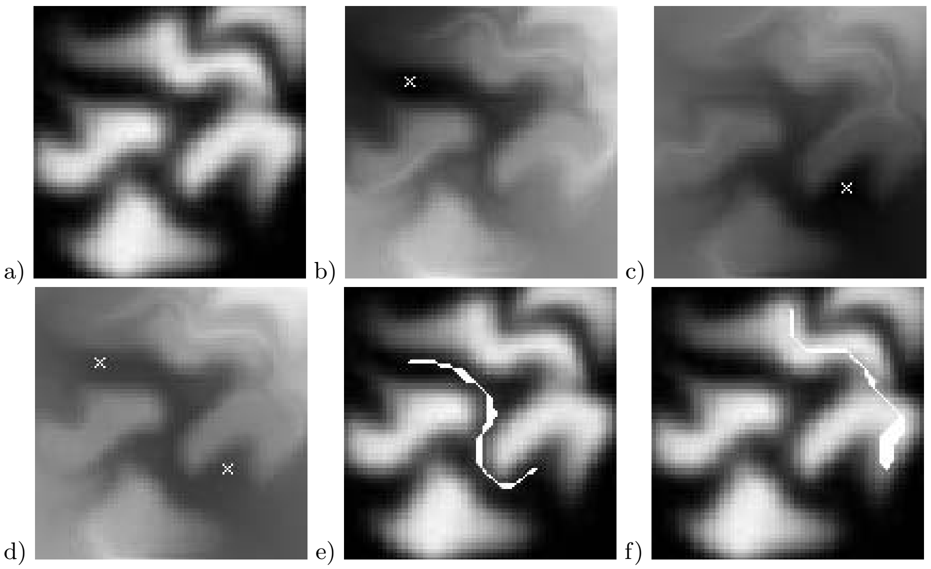


Fig. 2. a) Original image, b) distance from source point, c) distance from destination point, d) sum of distance images, e) route on original image, f) another route example: same original image but different source and destination point.

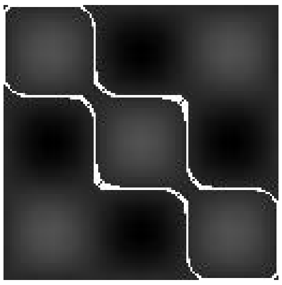
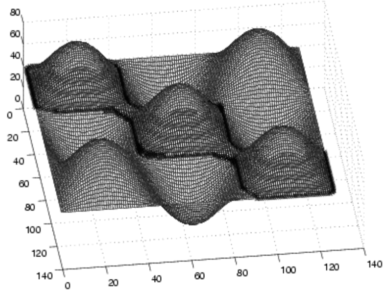
a)  b) 

Fig. 3. a) Shortest routes from corner to corner of an "eggbox" surface, b) 3D-visualization of the routes on the surface

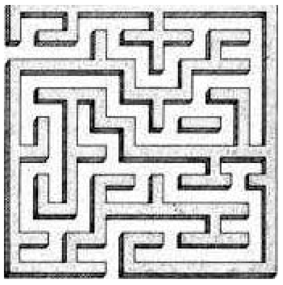
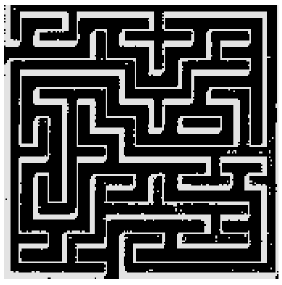
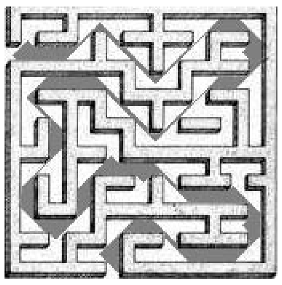
a)  b)  c) 

Fig. 4. a) Original labyrinth, b) the threshold segmented image, c) Route by DTOCS

6 Discussion

It can be seen in all the examples that the routes are typically more than one pixel wide, which means that there are actually several optimal paths. For example in figure 2 f) the lower end of the path can proceed on the right of the wide path segment, or the left, or anywhere in between. Getting a pixel-to-pixel description of a single optimal path is a question for future research, but in many applications visualizing all paths can be even more useful than finding one optimal path. For example, the shortest path(s) found between two cities on a terrain map could directly be used as a plan for a new highway connecting the cities. A similar map could be used in orienteering to find the easiest path through varying terrain. If for some reason none of the shortest paths found are feasible, the algorithm can easily be modified to show the second best path(s) as well. The gray-level values on the map do not have to correspond directly to height. For example, difficult terrains (e.g. mountains or swamps) can be marked with larger gray-level values to avoid them, both in orienteering and road planning.

The labyrinth solver presented in figure 4 may seem a toy, but it could have some practical applications, for example finding optimal routes through a city for a car or pedestrian. A city roadmap could be edited to include buildings and other obstacles as very high areas (i.e. labyrinth walls), and then use the algorithm to find the best route. Labyrinth or maze solving algorithms are also used for example in design of circuit board layouts.

The Route DTOCS is also a practical tool for analyzing distance transformation algorithms. The route in figure 4 c) may not be intuitively optimal. It seems to be too wide in some locations, i.e. the shortest route can either go straight along a wall segment, or make a seemingly extra 90° corner. Both routes are, however, correct according to the chessboard distance definition, where the distance between diagonal neighbors is the same as between horizontal and vertical neighbors. This is not a big problem when calculating local distances, but as the Route DTOCS calculates global distances across the whole image, the approximation error accumulates. In future works we will improve the shortest route algorithm by using more accurate distance definitions, for example the piecewise Euclidean distance of the Weighted Distance Transform on Curved Space (WDTOCS presented in [9]). Also the 3×3 and 5×5 masks with different integer approximations of distances to neighboring pixels presented in [1] could be applied.

References

1. Borgefors, G.: Distance Transformations in Digital Images. Computer Vision, Graphics, and Image Processing, 34 (1986), 344–371
2. 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
3. Kimmel, R., 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
4. 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
5. Piper J., Granum E.: Computing Distance Transformations in Convex and Non-Convex Domains. Pattern Recognition, vol 20, no 6 (1987) 599–615
6. Saab Y., 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
7. Toivanen, P.: Convergence properties of the Distance Transform on Curved Space (DTOCS). Proc. of Finnish Signal Processing Symposium (1995) 75–79
8. Rosin, P., West, G.: Salience Distance Transforms. Graphical Models and Image Processing, vol 56, no 6 (1995) 483–521
9. Toivanen, P.: New geodesic distance transforms for gray-scale images. Pattern Recognition Letters, 17 (1996) 437–450