Occlusion points propagation geodesic distance transformation

Conference Paper *in* Proceedings / ICIP ... International Conference on Image Processing · January 2003

DOI: 10.1109/ICIP.2003.1246973 · Source: DBLP

4 authors:

Rubén Cárdenes
HENSOLDT Sensors
71 PUBLICATIONS **676** CITATIONS
SEE PROFILE

Simon K Warfield
Harvard University and Boston Children's Hospital
614 PUBLICATIONS **21,906** CITATIONS
SEE PROFILE

Elsa M. Macías
Universidad de Las Palmas de Gran Canaria
75 PUBLICATIONS **316** CITATIONS
SEE PROFILE

Juan Ruiz-Alzola
Universidad de Las Palmas de Gran Canaria
94 PUBLICATIONS **1,229** CITATIONS
SEE PROFILE

OCCLUSION POINTS PROPAGATION GEODESIC DISTANCE TRANSFORMATION

Rubén Cárdenes[1,3]*, Simon K. Warfield*[2]*, Elsa Macías*[1] *and Juan Ruiz-Alzola*[2,3]

[1]Dept. Ingeniería Telemática, Universidad de Las Palmas de GC, SPAIN
[2]Harvard Medical School and Brigham and Women's Hospital, Dept. Radiology, USA
[3]Medical Technology Center, Univ. Las Palmas GC & Gran Canaria Dr. Negrín Hospital, SPAIN
E-mail:ruben@ctm.ulpgc.es,warfield@bwh.harvard.edu,elsa@dit.ulpgc.es,jruiz@ctm.ulpgc.es

ABSTRACT

We propose a new approach to compute geodesic distance transformations in arbitrary 2D and 3D domains. The distance transformation proposed here is robust and has proved to have a computational complexity linear in the domain size. Our scheme is based on a new technique which we call occlusion points propagation, and with a higher accuracy than other geodesic distance transformations proposed before. We validate the algorithm with a set of synthetic domains, and we also make comparisons with two similar algorithms called B_d-geodesic distance transformation and B_d-geodesic distance transformation with circular propagation.

1. INTRODUCTION

The distance transformation (DT) of a binary image with object and non object pixels, assign to every pixel, the distance to the nearest object pixel. There are several implementations for computing the DT efficiently, most of them taking advantage of the fact that distances vary smoothly in the domain, in order to deduce the value of the map in one pixel from the values of the map around it. Thus many DT algorithms are based on mask propagations, like in Rosenfeld [1] and Borgefors [2]. In the paper presented by Verwer [3], a new approach is proposed, which consists of an ordered propagation, from the objects to the rest of the image. With this idea, several DT algorithms have been developed, for example see Ragnemalm [4]. In the work presented by Piper et al. [5], the DT computation is extended to non convex domains using a geodesic metric. In this paper we propose a new DT algorithm to compute distance maps in arbitrary 2D and 3D convex and non convex domains. With this scheme, distance maps can be computed even in domains with obstacles and corners, using a geodesic metric different from the usual Euclidean metric. The most interesting approach proposed before is that of Cuisenaire [6], who proposed an ordered propagation geodesic DT called B_d-geodesic DT, based on a B_d-geodesic metric. We will implement a new metric that makes our scheme more efficient and accurate. Our method is called *occlusion points propagation geodesic distance transformation (OPPGDT)*, whose key feature is the detection of occlusion points in a domain from a selected point of view. This makes it possible to know where a corner or obstacle is located, in order to start a propagation front from there.

The first author is funded by a FPU grant from the University of Las Palmas de Gran Canaria. This work has been partially supported by the Spanish Ministry of Science and Technology and European Commission, co-funded grant TIC-2001-38008-C02-01

2. GEODESIC DISTANCE TRANSFORMATION AND OCCLUSION POINTS

The general definition of geodesic distance is as follows

Definition 1 *Suppose $P = \{p_1, p_2, ..., p_n\}$ is a path in a connected domain between pixels p_1 and p_n, i.e. p_i and p_{i+1} are connected neighbors for $i \in \{1, 2, ..., n-1\}$ and p_i belongs to the domain for all i. The geodesic distance between two pixels p_1 and p_n is defined as the length of the shortest path from p_1 to p_n where the path length $l(P)$ is defined as*

$$l(P) = \sum_{i=1}^{n-1} d_N(p_i, p_{i+1}) \quad (1)$$

i.e., the sum of the neighbor distances d_N between adjacent points in the path.

The simplest implementation of this metric is the geodesic version of the city block metric used in [5]. Some other metrics, such as the Chamfer metric, have been used for example in [3, 5], but they are coarse approximations to the geodesic version of the Euclidean DT. A more recent work presented by Cuisenaire [7] proposes a different definition of geodesic distance, which he called the B_d-geodesic distance, such that the distance between two pixels $\vec{p}$ and $\vec{q}$ is the Euclidean distance if $\vec{q}$ belongs to a ball B_d centered at $\vec{p}$ and radius d, as far as there is a path in the domain between both points. Note that for non convex domains the segment providing the Euclidean distance is not always completely included in the domain.

In this paper we propose to use a new definition of geodesic distance, that we will call the *occlusion points geodesic distance*. First of all we start defining the occlusion points.

Definition 2 *We define an occlusion point r with respect to s, in a domain $\mathcal{M}$, as the nearest point from s such that given $\epsilon > 0$, it does not exist a straight path included in $\mathcal{M}$ that joins r and s, but there exists a straight path included in $\mathcal{M}$, between s and a point x such that $x \in B_\epsilon(r)$, where $B_\epsilon(r)$ is a ball centered at r of radius ϵ.*

The discretization of this definition is straightforward changing the ball $B_\epsilon(r)$ by the two dimensional neighborhood of size 8: $N_8(r)$. With this definition, we can define the new geodesic distance as follows

Definition 3 *We define the occlusion points geodesic distance between p_1 and p_n, in a domain $\mathcal{M}$ as the sum of distances of equation 1, if the path $P = \{p_1, p_2, ..., p_n\}$ is such that p_i are occlusion points with respect to p_{i-1} $\forall\ i \in \{2, ..., n-1\}$, and $n > 2$ or if p_1 is directly connected with p_2 with a straight path included in $\mathcal{M}$ if $n = 2$, and the distances $d_N(p_i, p_{i+1})$ are Euclidean.*

This definition means that the geodesic path between a point p_1 to the point p_n, is a straight line if they are visually connected in the domain, or a chain of segments through the occlusion points if they are not visually connected. The geodesic path is like a tensed string that joins p_1 and p_n and restricted to the domain, see fig 1.

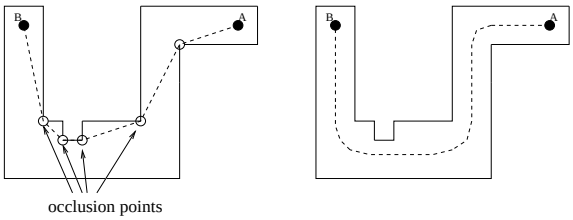

Fig. 1. Shortest path for the OPPGDT (left), and shortest path for the B_d-geodesic DT (right) between points A and B in a two dimensional non convex domain

The key point in the algorithm described here is how to find the occlusion points, through which the geodesic path goes on.

One efficient algorithm to implement usual geodesic distance transformations is that of Verwer et al. [3]. It scans the pixels in order of increasing distance by bucket sorting the pixels in the propagation front. In our method we propagate the distances to the nearest object as well as its coordinates, until the whole domain is visited. Propagations are performed by means of morphological dilations that, at each step increase in one unit the distance to the starting object, (distance zero). For this purpose, we use two lists: *list1*, stores the elements that are currently being propagated, and *list2* stores the elements to be propagated in the next dilation. For this reason, when we are dilating the elements corresponding to distance d_1, we expect to find elements placed at a distance $d_2 > d_1$, so if we find a point at distance $d_2 \leq d_1$, there is an anomaly corresponding to an obstacle or corner, i.e. an occlusion point appears. This idea of searching for an anomaly in the propagation front, in order to find an occlusion point, can be addressed using the following property.

Property 1 *Let $\mathcal{P}_d(o) = \{p | d - 0.5 < dist_e(p, o) \leq d + 0.5; d \in I\!N\}$ be the set of pixels whose centers are at an Euclidean distance $d \pm 0.5$ from an object o in a 2D grid, and let S be the dilation of $\mathcal{P}_d(o)$. The minimum structure element of the dilation of $\mathcal{P}_d(o)$ in a convex and free obstacle domain, such that $\mathcal{P}_{d+1}(o) \subset S$, is N_8.*

Proof. In order to obtain the set $\mathcal{P}_{d+1}(o)$ by means of a dilation, it is necessary to dilate with a structure element that completely contains a ball of radius 1. The minimum structure element that accomplishes this is N_8.

This property holds in a 3D grid for a 26 sized neighborhood (N_{26}) and it is necessary to find an occlusion point with respect to an object in a non convex domain.

3. THE ALGORITHM

For every pixel $\vec{p}$ in the domain we store its coordinates $\vec{p} = (p_x, p_y)$, the distance to the initial object from the nearest occlusion point it comes from, $d_{obj}(\vec{p})$, and the coordinates of the initial object or the nearest occlusion point it comes from $\vec{r}_{obj}(\vec{p}) = (x_{obj}(\vec{p}), y_{obj}(\vec{p}))$. The algorithm initializes the distance map D to -1, and puts the object points O in *list1*. Then it starts to extract and dilate consecutively the elements of *list1*, putting the new reached pixels in *list2*. When the *list1* is completely empty we swap *list1* and *list2* and repeat the process until both lists are empty, i.e. until all the pixels of the domain are processed.

The occlusion points appear when the propagation front reaches obstacles, or corners in the domain. We detect such occlusion points by means of *property 1*. If we dilate a pixel $\vec{p}$ with N_8, that belongs to the propagation front at distance d, and reach a pixel $\vec{q}$ at distance less or equal than d, then *property 1* is not accomplished, which means that the domain is non convex, so the pixel $\vec{q}$ is an occlusion point. In our method, once we detect an occlusion point, it automatically becomes a new object and a new propagation front starts from it. This is carried out propagating the occlusion point coordinates instead of the initial object coordinates from this point. For this reason, when a new pixel $\vec{p}$ is reached by a pixel $\vec{q}$, the new distance is computed as follows

$$d_{new}(\vec{p}) = dist_e(\vec{p}, r_{obj}(\vec{q})) + d_{obj}(\vec{q}) \qquad (2)$$

the Euclidean distance from $\vec{p}$ to the nearest occlusion point of $\vec{q}$, plus the distance from that occlusion point, to the initial object. For this reason, for every pixel, we propagate its nearest occlusion point coordinates, and the distance from this occlusion point to the initial object. The pseudo-code is as follows

Algorithm 1 *Occlusion points propagation geodesic distance transformation*

Input: N object points $O(i)$ from where we want to compute the geodesic DT, and the 2-dimensional domain $\mathcal{M}$ where the DT is restricted.
Output: the geodesic distance transformation D from the objects restricted to $\mathcal{M}$.

Initialize D and d_{obj}, for every pixel and put objects in $list1$
$d = 0$
while (list1 is not empty and list2 is not empty) **do**
  while (list1 is not empty) **do**
    get $(\vec{p}, d_{obj}(\vec{p}), \vec{r}_{obj}(\vec{p}))$ from list1
    propagate $(\vec{p}, d_{obj}(\vec{p}), \vec{r}_{obj}(\vec{p}), d)$
  end while
  $d = d + 1$
  swap(list1,list2)
  if (list-aux is not empty)
    append list-aux to list1
    empty list-aux
  end if
end while
procedure propagate$(\vec{p}, d_{obj}(\vec{p}), \vec{r}_{obj}(\vec{p}), d)$
  for all $\vec{n} \in N_8$ **do**
    if $(\vec{p} + \vec{n} \in \mathcal{M}$ and $D(\vec{p} + \vec{n}) = -1)$
      $d_{new} = dist_e(\vec{p} + \vec{n}, \vec{r}_{obj}(\vec{p})) + d_{obj}(\vec{p})$
      $d_{int} = round(d_{new})$
      if $(d_{int} = d + 1)$
        put $(\vec{p} + \vec{n}, d_{obj}(\vec{p}), \vec{r}_{obj}(\vec{p}))$ in list2
        $D(\vec{p} + \vec{n}) = d_{new}$
      end if
      if $(d_{int} \leq d)$ (occlusion point appears)
        put $(\vec{p} + \vec{n}, d_{new}, \vec{p})$ in list-aux
        $d_{obj}(\vec{p}) = d_{new}$
        $D(\vec{p} + \vec{n}) = d_{new} + dist_e(\vec{p} + \vec{n}, \vec{p})$
      end if
    end if
  end for
end procedure

Notice the importance of dilating with N_8, if we dilate with N_4, we are not able to find the occlusion points in the propagation front, due to *property 1*.

4. COMPUTATIONAL COMPLEXITY AND MEMORY LOAD

The computational complexity comes from the number of distance calculations in this algorithm. We need to calculate a new distance every time a new pixel is reached, i.e. the number of distance computations is m, the number of pixels in the domain $\mathcal{M}$. Thus the algorithm will have a computational complexity of order $O(m)$. There are a number of additional distance calculations corresponding to the appearance of occlusion points, but they are, in general, negligible with respect to m. The computational complexity of our approach is of the same order than the B_d-geodesic DT by circular propagation, but our approach is a little bit faster due to the number of comparisons made at every pixel. The number of comparisons for our approach is four for every pixel, and the number of comparisons by the B_d-geodesic DT by circular propagation is six for every new pixel. Both algorithms are one order of magnitude better than the B_d-geodesic DT.

The memory load is almost the same for both algorithms B_d-geodesic DT by circular propagation and OPPGDT, because both of them store for every pixel two vectors of integers and a float distance, and they manage two lists for the propagation front. In the case of the B_d-geodesic DT, the pixel storage is also the same, but the memory load is higher because it manages a number of buckets equal to the radius of the ball B_d, and it is necessary to allocate more memory for the bucket sorting propagation.

5. RESULTS

We have tested the algorithm in a 2D synthetic non convex domain, embedded in a 256x256 image with 22214 pixels and two up-sampled domains, one of size 512x512, with a domain size of 88856 pixels, and the other of size 1024x1024 with a domain size of 355424 pixels. Figure 3 a) and b) shows two geodesic DT for a single object in the 2D synthetic domain, represented with a cyclic colormap to give a better visualization of the distance transformation. Every iso-distances curve have a different color than its neighbors. Figure 3 c) and d) shows the same but using a grayscale colormap, where low distances are displayed darker and high distances lighter.

Table 1 shows execution times for the three geodesic DT algorithms, and for different domain sizes. The execution times has been measured in a SUN-Ultra 10 workstation with an Ultra-SPARC II 440 MHz processor and 512 MB RAM. Notice that the first two schemes have similar execution times, but our method is always faster, and the last scheme is always an order of magnitude slower than the others. Notice also that the times in the first two schemes increase linearly with the number of points in the domain, showing a computational complexity of $O(m)$ for both algorithms.

We show in figure 2 the maximum distance reached by the B_d-geodesic DT for different ball sizes, compared with the distance reached by the occlusion points geodesic DT, in the 2D synthetic domain of figure 3. Notice that the max-

geodesic DT	number of pixels	CPU time
occlusion points propagation	22214	0.0678
	88856	0.2740
	355424	1.1908
B_d by circular propagation	22214	0.0778
	88856	0.3178
	355424	1.3782
B_d	22214	0.3755
	88856	4.2081
	355424	39.3589

Table 1. Execution times in seconds for the three different geodesic DT algorithms, in the three experiments carried out

imum distance for our algorithm is always lower. We use ball size values up to 20 because higher values are not a good choice in order to follow the obstacles' shape.

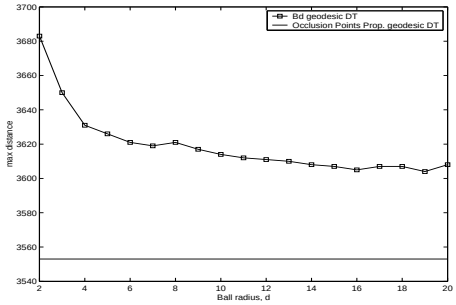

Fig. 2. Maximum distance reached vs ball radius (d) in the B_d-geodesic DT, compared with maximum distance in the OPPGDT

6. CONCLUSIONS

We have proposed here a new geodesic DT algorithm which is faster, and more accurate than any other geodesic DT proposed previously, such as the B_d-geodesic DT, which depends on the ball size parameter d, unlike our algorithm which does not depend on any parameter. The occlusion points paradigm allows our geodesic DT to work in connected convex and non convex 2D and 3D domains. When the domain is convex, our DT behaves like an usual Euclidean DT, and is more accurate than the B_d geodesic DT because the B_d metric have more errors in obstacle free domains see [7].

We have demonstrated that our geodesic metric is better than the B_d geodesic metric, as shown in figure 2 where the maximum distance reached by our method is always lower than the B_d-geodesic DT, for reasonable values of d, where the DT can still follows the domain's shape.

The computational complexity of our approach is of order $O(m)$, which is one order of magnitude better than the

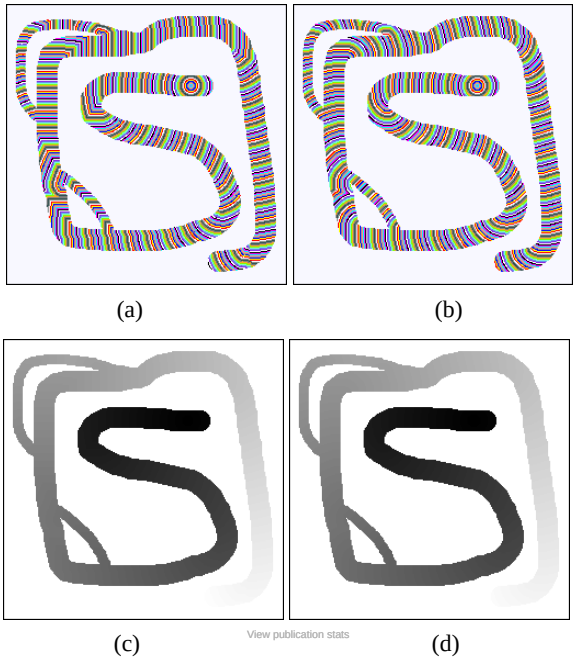

Fig. 3. OPPGDT coded with a cyclic colormap (a) and with a grayscale map (c), B_d-geodesic DT coded with a cyclic colormap, for $d = 6$ (b), and with a grayscale map, (d)

B_d-geodesic DT, and is slightly better than the B_d-geodesic DT by circular propagation, because we perform two less comparison operations per pixel than the B_d-geodesic DT by circular propagation.

7. REFERENCES

[1] A. Rosenfeld and J.L. Pfaltz, "Distance functions on digital pictures," *Pattern Recognition*, vol. 1(1), pp. 33–61, 1968.

[2] G. Borgefors, "Distance transformations in arbitrary dimensions," *Computer Vision, Graphics and Image Processing*, vol. 27, pp. 321–345, 1984.

[3] B.H. Verwer, P.W. Verbeek, and S.T. Dekker, "An efficient uniform cost algorithm applied to distance transforms," *IEEE Transactions on Pattern Analysis an Machine Intelligence*, vol. 11(4), pp. 425–429, 1989.

[4] I. Ragnemalm, "Neighborhoods for distance transformations using ordered propagation," *CVGIP, Image Understanding*, vol. 56, no. 3, pp. 399–409, 1992.

[5] J. Piper and E. Granum, "Computing distance transformations in convex and nonconvex domains," *Pattern Recognit*, vol. 20(6), pp. 599–615, 1987.

[6] O. Cuisenaire and B. Macq, "Fast k-nn classification with an optimal k-distance transformation algorithm," in *10th European Signal Processing Conf.*, Tampere, Finland, 2000, pp. 1365–1368.

[7] O. Cuisenaire, *Distance Transformations: fast algorithms and applications to medical image processing*, Ph.D. thesis, UCL, July 1999.