ELSEVIER

Pattern Recognition Letters 23 (2002) 1419–1426

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Curve skeletonization of surface-like objects in 3D images guided by voxel classification

S. Svensson [a,*], I. Nyström [b], G. Sanniti di Baja [c]

[a] *Centre for Image Analysis, Swedish University of Agricultural Sciences, Lägerhyddvägen 17, 75237 Uppsala, Sweden*
[b] *Centre for Image Analysis, Uppsala University, Uppsala, Sweden*
[c] *Istituto di Cibernetica, National Research Council of Italy (CNR), Pozzuoli (Naples), Italy*

Abstract

Skeletonization is a way to reduce dimensionality of digital objects. Here, we present an algorithm that computes the curve skeleton of a surface-like object in a 3D image, i.e., an object that in one of the three dimensions is at most two-voxel thick. A surface-like object consists of surfaces and curves crossing each other. Its curve skeleton is a 1D set centred within the surface-like object and with preserved topological properties. It can be useful to achieve a qualitative shape representation of the object with reduced dimensionality. The basic idea behind our algorithm is to detect the curves and the junctions between different surfaces and prevent their removal as they retain the most significant shape representation.

Keywords: Curve skeleton; Topology preservation; Shape representation; Volume image

1. Introduction

Skeletonization is a way to reduce dimensionality of digital objects. For reduction of objects in 2D images to 1D skeletons (see, e.g., Suen and Wang, 1994; Sanniti di Baja and Thiel, 1996; Svensson et al., 1999a); from 3D objects to 2D surface skeletons (see, e.g., Bertrand, 1995; Borgefors et al., 1999; Sanniti di Baja and Svensson, 2000b); and from 3D objects to 1D curve skeletons (see, e.g., Bertrand and Aktouf, 1994; Palágyi and Kuba, 1999; Saha et al., 1997). For an open surface in a 3D image, skeletonization results in a curve, as in (e.g., Nyström et al., 2001a). All above papers regard discrete approaches to skeletonization. Alternative approaches can be found, e.g., in (Attali and Montanvert, 1997; Leymarie and Kimia, 2001).

We follow the discrete approach and, here, we deal with open surfaces or rather surface-like objects, i.e., objects that in one of the three dimensions are at most two-voxel thick. Examples of surface-like objects are the sets resulting after a skeletonization process from 3D solid objects to their surface skeletons. We are interested in computing the curve skeletons of surface-like objects. The curve skeleton is a 1D set centred within the object, with the same topological properties. Of course, the original object can not be recovered

* Corresponding author. Tel.: +46-18-4713465; fax: +46-18-553447.
E-mail address: stina@cb.uu.se (S. Svensson).

0167-8655/02/$ - see front matter
PII: S0167-8655(02)00102-2

starting from its curve skeleton, but the curve skeleton is still useful to achieve a qualitative, and to a certain extent quantitative, shape representation of the object with reduced dimensionality. It is particularly important to identify the voxels constituting the curve skeleton in such a way that shape information is retained as much as possible. In fact, the bottleneck in the practical use of the curve skeleton is that reduction of dimensionality unavoidably causes loss of shape information.

A crucial point in curve skeletonization is the detection of the end-points, i.e., the voxels that in the obtained curve skeleton are delimiting peripheral branches. When, during curve skeletonization, part of the surface is transformed into a 1D set, the voxels delimiting this 1D set could be removed without altering topology. However, their removal would cause unwanted shortening and thereby important shape information would be lost. End-point detection criteria based on the number (and possibly the position) of neighbouring voxels are blind in the sense that it is not known a priori which end-points (and, hence, which branches) the curve skeleton will have. In fact, peripheral parts of the surface could consist of identical local configurations of object voxels that are differently oriented. Thus, depending on the order in which voxels are checked for removal, some of these configurations may originate end-points and, hence, skeleton branches, while other configurations may be completely removed. More reliable end-point detection criteria are based on geometrical properties, e.g., end-points could be detected in correspondence with convexities on the border of the object.

We present an algorithm to compute the curve skeleton from a surface-like object and use geometrical information to automatically ascribe to the curve skeleton the voxels that will play the role of end-points in the curve skeleton.

A surface-like object consists of surfaces and curves crossing each other. The basic idea behind our algorithm is to detect the curves and the junctions between different surfaces and prevent their removal as they retain the most significant shape representation. This would avoid unwanted shortening of future skeletal branches without need of any specific end-point detection criterion. To implement our idea, a classification of the voxels belonging to the surface-like object is necessary to distinguish *junction*, *inner*, *edge*, and *curve* voxels, see Section 2. In Fig. 1, a surface-like object, top, and its resulting classes, bottom, are shown. All curve voxels have to be ascribed to the curve skeleton. Among junctions, we distinguish branches, i.e., junctions having at least one free tip, and loops. All branches that are peripheral junctions carry shape information and have to be ascribed to the curve skeleton. In the example shown in Fig. 1, the only branch that has not to be ascribed to the curve skeleton is the non-peripheral junction pointed out by the arrow. The junctions

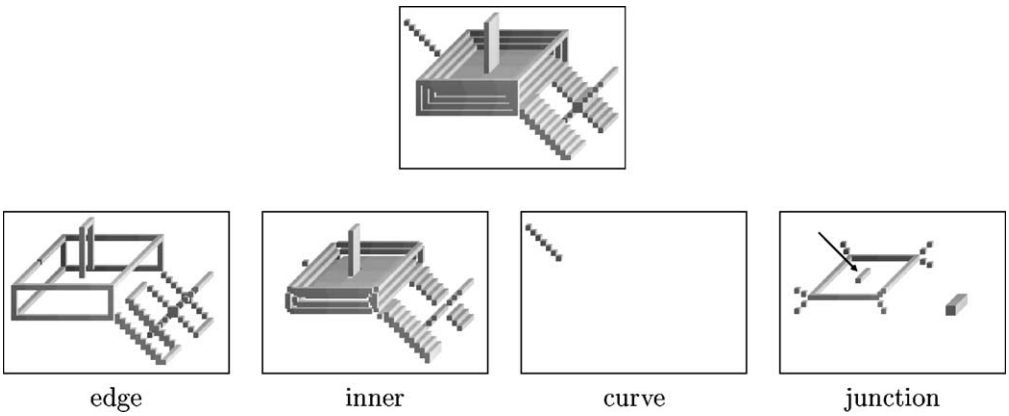

Fig. 1. A surface-like object, top, with its classification, bottom.

constituting a loop, i.e., delimiting a surface part, will, if not removed, prevent the skeletonization process to produce a curve skeleton.

Our algorithm computes the curve skeleton in two steps, both based on iterated edge voxel removal. During the first step, all curve and junction voxels found in the original surface-like object are prevented from being removed. During the second step, also voxels initially classified as junction voxels are candidate for removal, if they are now classified as edge voxels (see Section 4 for details).

The algorithm has been introduced and outlined in (Nyström et al., 2001a). The aspects we have investigated in more detail in this paper are the description of the classification algorithm used to distinguish the various kinds of voxels in the surface-like object (Sanniti di Baja and Svensson, 2000a), and the use of a simplification technique to remove noisy branches from the surface-like object before applying skeletonization (Borgefors et al., 2000). A brief description of the simplification is given in Section 3. The performance of our curve skeletonization algorithm can be found in Section 5.

2. Classification

We refer to digital volume images consisting of surface-like objects and background. The 26-connectedness is chosen for the surface-like object S and the 6-connectedness for the background.

Classification of the voxels in a surface was suggested in (Malandain et al., 1993; Saha and Chaudhuri, 1994). A voxel is classified after investigating its $3 \times 3 \times 3$ neighbourhood. That classification works for an "ideal" surface, i.e., a surface which is one-voxel thick everywhere. However, even ideal surfaces may cross each other in such a way that they produce a two-voxel thick junction whose voxels could not be identified as junction voxels by the criteria suggested by Malandain et al. (1993) or Saha and Chaudhuri (1994). In general, that classification fails when applied to surface-like objects that are two-voxel thick, e.g., surface skeletons of 3D objects having regions that are an even number of voxels thick.

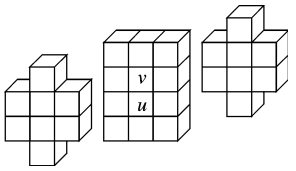

Fig. 2. The 27-neighbourhood$_y$ of v.

The 26-neighbours of any voxel v are called face, edge, and point neighbours, depending on whether they share a face, an edge, or a point with v. The $3 \times 3 \times 3$ set centred on v is called the 26-neighbourhood, to take into account that this set includes, besides v, all the 26-neighbours of v. If the eight point neighbours of v are disregarded in the 26-neighbourhood, the obtained set is called the 18-neighbourhood. Since we are dealing with surface-like objects, i.e, objects that can be two-voxel thick, we also consider $4 \times 3 \times 3$, $3 \times 4 \times 3$, and $3 \times 3 \times 4$ sets. In particular, if the eight voxels in the corners of the $4 \times 3 \times 3$ ($3 \times 4 \times 3$, $3 \times 3 \times 4$) set are disregarded, the obtained set is called the 27-neighbourhood$_x$ (27-neighbourhood$_y$, 27-neighbourhood$_z$). See Fig. 2, where the 27-neighbourhood$_y$ of v is shown, and the two voxels v and u are face neighbours in the y-direction. We call u the face neighbour of v in the x-, y-, or z-direction.

For any voxel v,

- N^{26} denotes the number of 26-connected object components in the 26-neighbourhood of v,
- $\overline{N}_{\mathrm{f}}^{18}$ denotes the number of 6-connected background components in the 18-neighbourhood of v, having v as a face neighbour,
- $\widehat{N}_{\mathrm{f}}^{27}$ denotes the number of 6-connected background components in the 27-neighbourhood$_{x,y,z}$ of v, having v or u as a face neighbour.

We use the computationally convenient algorithm introduced by Borgefors et al. (1997) for computing N^{26} and $\overline{N}_{\mathrm{f}}^{18}$, suitably extended to deal also with a larger neighbourhood necessary for the computation of $\widehat{N}_{\mathrm{f}}^{27}$.

A saddle$_{x,y,z}$ is a set of four voxels aligned along the x-, y-, or z-direction, where the two internal voxels belong to S and the two external voxels to the background.

When working with surface-like objects, to compute the curve skeleton we need to distinguish the following classes: inner, curve, edge, and junction voxels. We define the border of S as the set including all curve and edge voxels. The following criteria are used to obtain the classification.

Voxels for which

(1) $N^{26} \geqslant 2$ and $\overline{N}_{\mathrm{f}}^{18} \geqslant 1$ are classified as curve voxels.
(2) $N^{26} = 1$ and $\overline{N}_{\mathrm{f}}^{18} > 2$ are classified as junction voxels.
(3) $N^{26} = 1$ and $\overline{N}_{\mathrm{f}}^{18} = 0$ are classified as junction voxels.
(4) $N^{26} = 1$ and $\overline{N}_{\mathrm{f}}^{18} = 2$ can be one-voxel thick inner voxels, or two-voxel thick junction voxels.
(5) $N^{26} = 1$ and $\overline{N}_{\mathrm{f}}^{18} = 1$ can be border voxels, two-voxel thick inner voxels, or two-voxel thick junction voxels.

For any voxel v satisfying condition (4) (condition (5)) further processing is necessary. The voxel v is checked to see whether it is in a saddle$_{x,y,z}$ with a voxel, u, also satisfying condition (4) (condition (5)). Each voxel, v, in a saddle$_{x,y,z}$ is thereafter checked in the 27-neighbourhood$_{x,y,z}$. Voxels for which

(6) $\widehat{N}_{\mathrm{f}}^{27} = 2$ are classified as inner voxels.
(7) $\widehat{N}_{\mathrm{f}}^{27} > 2$ are classified as junction voxels.
(8) $\widehat{N}_{\mathrm{f}}^{27} = 1$ are classified as border voxels (and can as such be either edge voxels or curve voxels).

The voxels satisfying condition (4) that are not in a saddle$_{x,y,z}$ are classified as inner voxels.

The voxels satisfying condition (5) that are not in a saddle$_{x,y,z}$ are classified as border voxels.

Finally, for all border voxels, i.e., voxels satisfying conditions (5) or (8), a decision can be taken on whether they are edge or curve voxels. Any border voxel v is classified as a curve voxel if

(9) it has all its neighbours in S classified as border voxels, or
(10) it has a neighbouring voxel, u, that is curve voxel, such that u does not have a neighbour that is an inner voxel.

The remaining border voxels are classified as edge voxels.

For the sake of completeness, we point out that a few number of special cases occurring in presence of junctions thicker than two voxels in the x-, y-, or z-direction have to be treated separately to get a classification consistent with the classification that would be achieved in case of an "ideal" surface. These cases have been exhaustively considered in (Sanniti di Baja and Svensson, 2000a). After the classification process, each voxel is identified as belonging to only one of the classes and there are no unclassified voxels.

3. Surface simplification

A surface-like object, S, with jagged borders might cause problems when computing the curve skeleton as the jaggedness causes the creation of a number of unnecessary curves. This is particularly true when S is the result of a surface skeletonization algorithm applied to 3D solid objects. The goal of surface simplification is to remove the jaggedness of the border of S while preserving the topology and without significantly altering its shape.

The border of S can be rather jagged due to the presence of a number of short curves, possibly two-voxel thick. The idea behind the simplification is to detect (peripheral) curves and remove them if they are shorter than a chosen threshold, without affecting the remaining curves. The threshold is set equal to the length of the longest curve we accept to remove. Obviously, a classification of the voxels of S is necessary before simplification. We base our simplification on the algorithm introduced in (Borgefors et al., 2000), but using the current, improved, classification.

After the voxels of S have been classified, to identify curves shorter than the threshold, we propagate along each curve a marker starting from the curve voxel(s) having inner neighbours. For simplicity, we denote as *connecting voxels* the voxels of the curve from which the marker is propagated. Each curve has from one to four connecting voxels, depending on the thickness of the curve. Propagation is done parallelwise, for a number

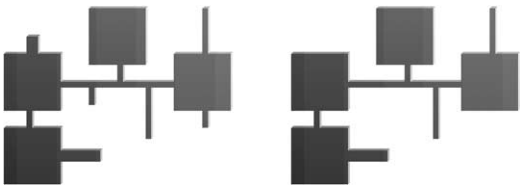

Fig. 3. A surface-like object, left, and the result after simplification, right.

of steps equal to the threshold. In this way, the voxels that are reached by the marker are exactly those at distance from the connecting voxels less than or equal to the maximal length a curve can have to be removed. Propagation according to the 26-connectedness is chosen since the object is 26-connected. At the end of propagation, curves whose voxels are all marked can be sequentially removed, provided that topology is not altered, i.e., provided that each curve voxel has $N^{26} = 1$ and $\overline{N}_{\mathrm{f}}^{18} = 1$ when visited. Removal is repeated until no further changes occur. In turn, curves whose voxels are not all marked are longer than the simplification threshold. Thus, they are kept.

We remark that also two-voxel thick short curves linking distinct parts of S are thinned to one-voxel thickness during simplification. Thus, to remove only peripheral short curves, while leaving all other curves untouched, a curve recovery process is performed. This restores all possibly deleted voxels in two-voxel thick linking curves, provided that they have a non-deleted curve voxel among their neighbours.

An example of the simplification of a surface-like object is shown in Fig. 3. All curves of length at most three voxels have been removed provided that the topology is not altered.

4. Curve skeletonization

The curve skeleton is computed in two steps using an iterative algorithm. For both steps, each iteration is divided into two subiterations respectively dealing with (i) detection of edge voxels and (ii) voxel removal by means of topology preserving removal operations. The only difference between the two steps regards the voxels that, at each iteration, are candidate for removal, i.e., the edge voxels.

In the first step, only voxels *initially* (i.e., on the surface-like object, S) classified as inner (or edge) voxels are checked when identifying the edge voxels. Voxels initially classified as inner voxels may, during the process, become edge voxels, if their neighbourhood has been suitably modified due to removal of neighbouring voxels.

We do not need any end-point detection criterion because (curve and) junction voxels are never checked to establish whether they have become edge voxels, iteration after iteration, so that no undesirable branch shortening is caused. On the contrary, this would have been the case if also the voxels initially classified as junction voxels were checked. In fact, voxels placed on the free tips of junctions could be classified as edge voxels and, as such, could be removed.

In the second step, also voxels initially classified as junction voxels are checked when identifying the edge voxels. Edge voxels that have been transformed into curves during the first step are automatically preserved from removal. The remaining voxels initially classified as junction voxels are interpreted as edge voxels, if their neighbourhood has been modified due to removal of neighbouring voxels.

Standard topology preserving removal operations, e.g., those described in (Svensson et al., 1999b), are sequentially applied in both steps. At the beginning of each iteration, the set of edge voxels is determined on which to perform removal. Edge detection and voxel removal are iterated until no more edge voxels are identified and possibly removed.

If the set of junctions of S does not include any loop, the curve skeleton is obtained directly after the first step. In this case, the second step terminates after the first subiteration, when the classification shows that no edge voxels can be found. The curve skeleton consists of the initial curve and junction voxels, as well as voxels necessary for connectedness among curves and junctions. In presence of loops in the set of junctions, the second step is effective as the first step has caused some junctions initially forming loops (see Fig. 1) to

become edge voxels. (Of course, the first step may have caused some junctions to be transformed into inner voxels, e.g., the junction voxels pointed out by the arrow in Fig. 1). In this way, the skeletonization can continue towards the innermost part of S.

Note that loops in the set of junctions do not necessarily correspond to tunnels in S, see again Fig. 1. Hence, their presence in the curve skeleton is not permitted as skeletonization is requested to be topology preserving. Thus, it is not possible to simply take all curves and junctions together with voxels necessary for connectedness to obtain a topologically correct curve skeleton.

The obtained curve skeleton is not necessarily one-voxel thick everywhere. Analogously to the term surface-like object the obtained set could be referred to as a *curve-like object*. Final reduction to one-voxel thickness could be achieved by resorting to standard thinning.

For the sake of completeness, we point out that S could have been reduced to one-voxel thickness before extracting the curve skeleton. One reason why we prefer not to do so, is that the resulting curve skeleton could be more than one-voxel thick anyway. This is the case, for instance, when computing the curve skeleton of a rectangle, one-voxel thick in one direction, but with its shortest side consisting of an even number of voxels. Hence, final thinning would be necessary in any case, if one-voxel thickness is considered of interest.

The result of curve skeletonization on the small example in Fig. 1 is shown in Fig. 4. The set obtained after the first step is shown to the left. It can be noted that some of the voxels classified as junction voxels in S are skeleton branches, while other junction voxels, e.g., those constituting the loop, prevent skeletonization to continue. In fact, voxels classified as junction voxels in S are not checked to verify if they have become edge voxels. Hence, they cannot be removed during the first step. During the second step, all voxels are checked to verify if they have become edge voxels, including those classified as junction voxels in S. Among those, the peripheral branches are classified as curve voxels and, hence, ascribed to the curve skeleton; those in the loop are classified as edge voxels and, hence, possibly removed; and those in the non-peripheral branch are classified as inner voxels. The set resulting after the second step is shown to the right.

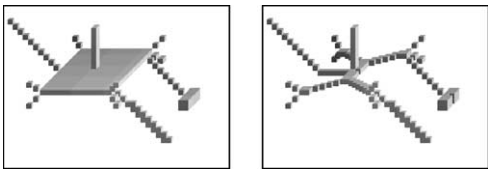

Fig. 4. Results after the first step, left and the second step, right, of curve skeletonization applied to the surface-like object shown in Fig. 1.

5. Performance

Projections of thin complex structures are hard to visualize in a descriptive way. In order to appreciate the performance at voxel level, we are showing the results of our algorithm on rather small synthetic objects.

The algorithm is first illustrated on two examples showing the importance of the simplification process. For both examples, the threshold for curve length is set to 3. In Fig. 5, top left, the

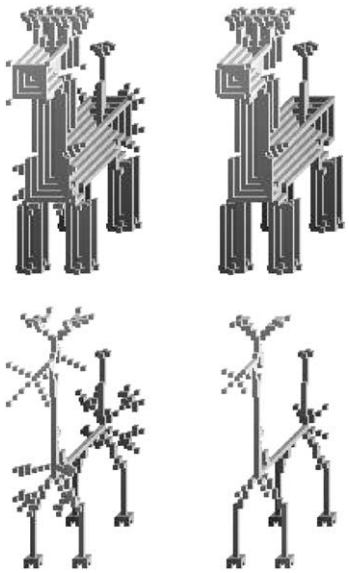

Fig. 5. Surface-like object before and after simplification, top. The corresponding curve skeletons, bottom.

surface skeleton resulting from a toy object, a dog, is shown. Some peripheral short curves have been created by the surface skeletonization method or by some noise on the border of the original object. If curve skeletonization is directly applied to this surface-like object, the set shown in Fig. 5, bottom left, is obtained. In turn, if a simplification of the surface-like object is computed before curve skeletonization, see Fig. 5, top right, a simpler curve skeleton, still reflecting the shape of the surface-like object, is obtained, see Fig. 5, bottom right.

In the same way as for the previous example, Fig. 6, top, shows the surface-like object before and after the simplification, while Fig. 6, bottom, the corresponding curve skeletons. This example is also interesting to point out that curve skeleton branches originate also from convexities on the border of the surface-like object. In fact, convexities like the ones in the example (angle less than 90°) during iterated voxel removal are shrunk to curves. Although these curves are very short as they consist of one or two voxels, they are prevented from removal since all curves are preserved by our algorithm.

Two other examples, Fig. 7, top, are surface-like objects, representing a chair and a cupboard,

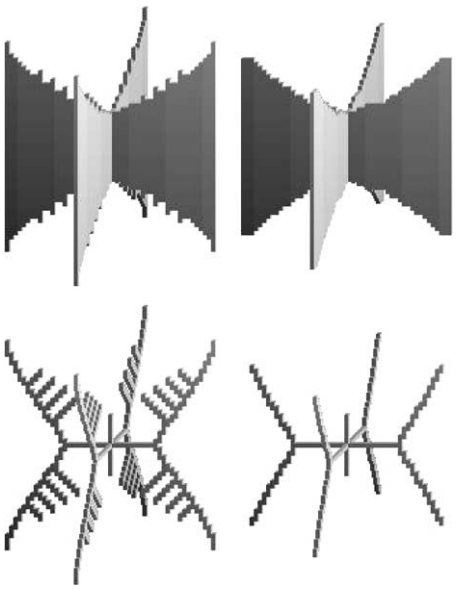

Fig. 6. Surface-like object before and after simplification, top. The corresponding curve skeletons, bottom.

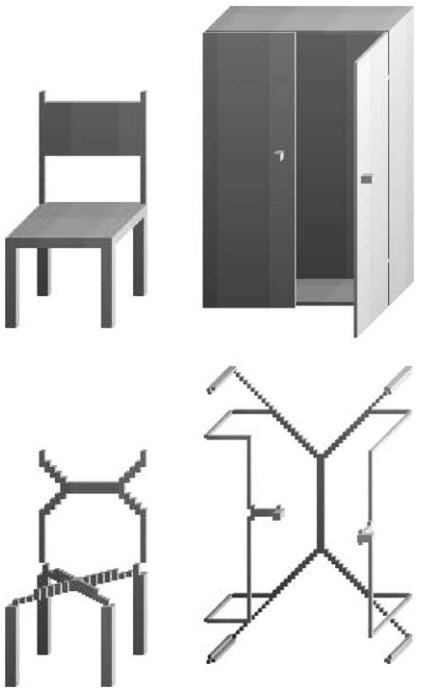

Fig. 7. Two surface-like objects, top, and their curve skeletons, bottom.

respectively. Both these objects are not everywhere one-voxel thick. In these cases, simplification is not applied since these surfaces are synthetic objects directly created and all the details have to be considered as significant. The resulting curve skeletons are shown in Fig. 7, bottom.

6. Conclusion

In this paper, we have presented an algorithm that computes the curve skeleton of a surface-like object in a 3D image. The algorithm is based on the classification of the surface-like object and in particular on the detection of curve and junction voxels. A simplification step, essential when working with real images, is also included to remove from the surface-like object short noisy peripheral curves that would otherwise lead to a curve skeleton with a complex structure so preventing its actual utilization. If desired, a final thinning can be applied to the curve skeleton to have one-voxel thickness everywhere.

A remarkable feature of the algorithm is that the end-points are automatically identified by preventing from removal curve and (some of the) junction voxels. Another important feature, particularly of interest when the surface-like object is the result from a surface skeletonization algorithm applied to a 3D solid object, is that our curve skeletonization algorithm equally works on one-voxel and two-voxel thick surfaces.

The computational cost of the curve skeletonization algorithm is a couple of minutes for complex real objects in images of size $128 \times 128 \times 128$ voxels. The main cost is due to the (non-optimized) classification process that is repeatedly used during curve skeletonization. A first use in a medical application is presented in (Nyström et al., 2001b).

Acknowledgements

We are thankful to Prof. Gunilla Borgefors, Centre for Image Analysis, Uppsala, Sweden, for useful discussions on skeletonization methods.

References

Attali, D., Montanvert, A., 1997. Computing and simplifying 2D and 3D continuous skeletons. Computer Vision and Image Understanding 67 (3), 261–273.

Bertrand, G., 1995. A parallel thinning algorithm for medial surfaces. Pattern Recognition Lett. 16, 979–986.

Bertrand, G., Aktouf, Z., 1994. A three-dimensional thinning algorithm using subfields. In: Melter, R.A., Wu, A.Y. (Eds.), Vision Geometry III. Proceedings of SPIE, Vol. 2356, pp. 113–124.

Borgefors, G., Nyström, I., Sanniti di Baja, G., 1997. Connected components in 3D neighbourhoods. In: Frydrych, M., Parkkinen, J., Visa, A. (Eds.), Proceedings of 10th Scandinavian Conference on Image Analysis (SCIA'97). Pattern Recognition Society of Finland, pp. 300–309.

Borgefors, G., Nyström, I., Sanniti di Baja, G., Svensson, S., 2000. Simplification of 3D skeletons using distance information. In: Latecki, L.J., Melter, R.A., Mount, D.M., Wu, A.Y. (Eds.), Vision Geometry IX. Proceedings of SPIE, Vol. 4117.

Borgefors, G., Nyström, I., Sanniti di Baja, G., 1999. Computing skeletons in three dimensions. Pattern Recognition 32 (7), 1225–1236.

Leymarie, F.F., Kimia, B.B., 2001. The shock scaffold for representing 3D shape. In: Arcelli, C., Cordella, L.P., Sanniti di Baja, G. (Eds.), Visual Form 2001. Lecture Notes in Computer Science, Vol. 2059. Springer-Verlag, pp. 216–228.

Malandain, G., Bertrand, G., Ayache, N., 1993. Topological segmentation of discrete surfaces. Internat. J. Comput. Vision 10 (2), 183–197.

Nyström, I., Sanniti di Baja, G., Svensson, S., 2001a. Curve skeletonization by junction detection. In: Arcelli, C., Cordella, L.P., Sanniti di Baja, G. (Eds.), Visual Form 2001. Lecture Notes in Computer Science, Vol. 2059. Springer-Verlag, pp. 229–238.

Nyström, I., Sanniti di Baja, G., Svensson, S., 2001b. Representing volumetric vascular structures using curve skeletons. In: Ardizzone, E., Di Gesù, V. (Eds.), Proceedings of 11th International Conference on Image Analysis and Processing (ICIAP 2001). IEEE Computer Society, pp. 495–500.

Palágyi, K., Kuba, A., 1999. A parallel 3D 12-subiteration thinning algorithm. Graphical Models and Image Processing 61, 199–221.

Saha, P.K., Chaudhuri, B.B., 1994. Detection of 3-D simple points for topology preserving transformations with application to thinning. IEEE Trans. Pattern Anal. Machine Intell. 16 (10), 1028–1032.

Saha, P.K., Chaudhuri, B.B., Majumder, D.D., 1997. A new shape preserving parallel thinning algorithm for 3D digital images. Pattern Recognition 30 (12), 1939–1955.

Sanniti di Baja, G., Svensson, S., 2000a. Classification of two-voxel thick surfaces: a first approach. Internal Report 19, Centre for Image Analysis, available from the authors.

Sanniti di Baja, G., Svensson, S., 2000b. Surface skeletons detected on the D^6 distance transform. In: Ferri, F.J., Iñetsa, J.M., Amin, A., Pudil, P. (Eds.), Proceedings of S+SSPR 2000: Advances in Pattern Recognition. Lecture Notes in Computer Science, Vol. 1876. Springer-Verlag, Berlin Heidelberg, pp. 387–396.

Sanniti di Baja, G., Thiel, E., 1996. Skeletonization algorithm running on path-based distance maps. Image Vision Comput. 14, 47–57.

Suen, C.Y., Wang, P.S.P. (Eds.), 1994. Thinning Methodologies for Pattern Recognition. Machine Perception and Artificial Intelligence, Vol. 8. World Scientific Publishing Co. Pte. Ltd.

Svensson, S., Borgefors, G., Nyström, I., 1999a. On reversible skeletonization using anchor-points from distance transforms. J. Visual Communication and Image Representation 10 (4), 379–397.

Svensson, S., Nyström, I., Borgefors, G., 1999b. Fully reversible skeletonization for volume images based on anchor-points from the D^{26} distance transform. In: Ersbøll, B.K., Johansen, P. (Eds.), Proceedings of the 11th Scandinavian Conference on Image Analysis (SCIA'99). The Pattern Recognition Society of Denmark, pp. 601–608.