Curve Skeletonization by Junction Detection in Surface Skeletons

Ingela Nyström[1], Gabriella Sanniti di Baja[2], and Stina Svensson[3]

[1] Centre for Image Analysis, Uppsala University
Lägerhyddvägen 17, SE-75237 Uppsala, SWEDEN
ingela@cb.uu.se
[2] Istituto di Cibernetica, National Research Council of Italy
Via Toiano 6, IT-80072 Arco Felice (Naples), ITALY
gsdb@imagm.cib.na.cnr.it
[3] Centre for Image Analysis, Swedish University of Agricultural Sciences
Lägerhyddvägen 17, SE-75237 Uppsala, SWEDEN
stina@cb.uu.se

Abstract. We present an algorithm that, starting from the surface skeleton of a 3D solid object, computes the curve skeleton. The algorithm is based on the detection of curves and junctions in the surface skeleton. It can be applied to any surface skeleton, including the case in which the surface skeleton is two-voxel thick.

1 Introduction

Reducing discrete structures to lower dimensions is desirable when dealing with volume images. This can be done by skeletonization. The result of skeletonization of a 3D object is either a set of surfaces and curves, or, if even more compression is desired and the starting object is a solid object, a set of only curves. In the latter case, the curve skeleton of the object is obtained. The curve skeleton is a 1D set centred within the object and with the same topological properties. Although the original object cannot be recovered starting from its curve skeleton, this is useful to achieve a qualitative shape representation of the object with reduced dimensionality.

There are two different approaches to compute the curve skeleton of a 3D object. One approach is to directly reduce the 3D object to its curve skeleton. See for example [1,5,7]. Another approach is to first obtain a surface skeleton from the 3D object. Thereafter, the curve skeleton can be computed from the surface skeleton. See for example [2,10,12]. For both approaches, maintenance of the topology is not too hard to fulfil as topology preserving removal operations are available. The more crucial point is the detection of the end-points, i.e., the voxels delimiting peripheral branches in the curve skeleton. These voxels could in fact be removed without altering topology, but their removal would cause unwanted shortening and thereby important shape information would be lost. Different end-point detection criteria can be used. Criteria based on the number (and possibly position) of neighbouring voxels are blind in the sense that it is not

C. Arcelli et al. (Eds.): IWVF4, LNCS 2059, pp. 229–238, 2001.

known a priori which end-points (and, hence, which branches) the curve skeleton will have. In fact, local configurations of object voxels, initially identical to each other, may evolve differently, due to the order in which voxels are checked for removal. Thus, the end-point detection criterion is sometimes fulfilled and sometimes not for identical configurations. Preferable end-point detection criteria are based on geometrical properties, i.e., end-points are detected in correspondence with convexities on the border of the object, or with centres of maximal balls. As far as we know, only blind criteria were used, when the first approach (object → curve skeleton) was followed. Therefore, we regard the second approach (object → surface skeleton → curve skeleton) as preferable, especially when a distance transform based algorithm is used in the object → surface skeleton phase. In fact, in this case the surface skeleton can include all the centres of maximal balls, [9]. This guarantees that end-points delimiting curves in the surface skeleton are automatically kept. The problem of not removing those end-points and correctly identifying other end-points during the surface skeleton → curve skeleton phase still needs to be carefully handled.

We present an algorithm to compute the curve skeleton from the surface skeleton and use geometrical information to ascribe to the curve skeleton voxels placed in curves and in (some of the) junctions, including voxels that will play the role of end-points in the curve skeleton.

A surface skeleton consists, in most cases, 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. This would automatically prevent unwanted shortening of curves and junctions without need of any end-point detection criterion. Indeed, we start from an initial classification of voxels in the surface skeleton, [8]. We distinguish *junction*, *inner*, *edge*, and *curve* voxels, see Fig.1. The border of the surface skeleton is the set including both edge and curve voxels. All curve voxels should be ascribed to the curve skeleton. Junctions shown in Fig.1 are placed in the innermost regions of the surface and should also be ascribed to the curve skeleton. However, junctions are not always in the innermost part of the surface. In fact, junctions may group in such a way that they delimit a surface in the surface skeleton. See Fig.2, where all junctions in the surface skeleton are shown to the right. Only junctions that could be interpreted as *peripheral* branches in the set of junctions should be kept in the skeleton, while junctions grouped into *loops* should not, as this would prevent the curve skeleton to be obtained. (Note also that loops in the curve skeleton correspond to tunnels in the object, and no tunnels exist in the surface skeleton shown in Fig.2.)

Our algorithm computes the curve skeleton from the surface 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 skeleton are always prevented from being removed. During the second step, voxels initially classified as junction voxels are prevented from removal, only if they are now classified as curve voxels; all voxels detected as edge voxels during this step are possibly removed. (Note

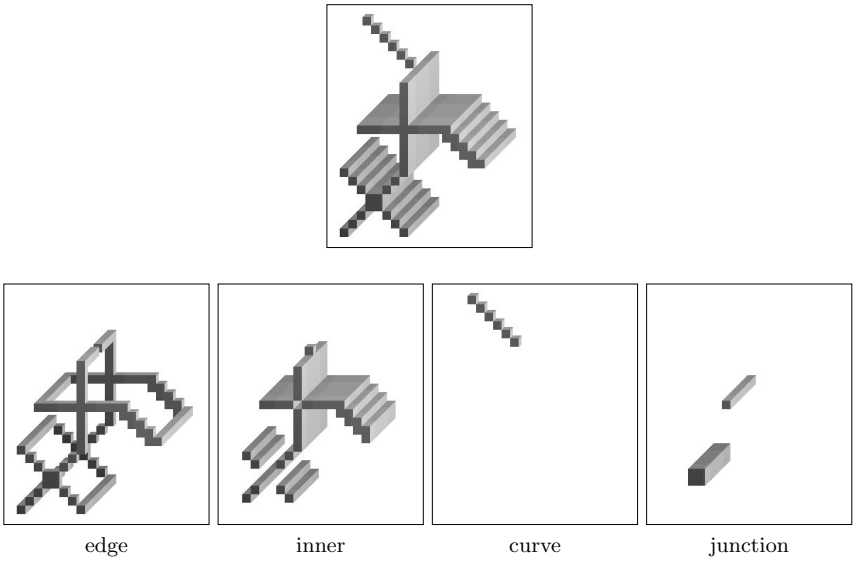

Fig. 1. A surface, top, with its classification, bottom.

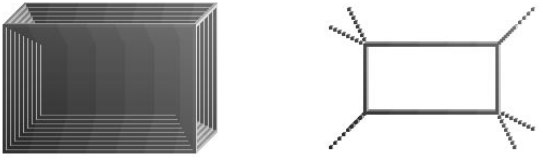

Fig. 2. A surface skeleton, left, and its junction voxels, right.

that also voxels that were classified as junction voxels during the classification done on the original surface skeleton are now possibly removed.)

The algorithm outlined above can be applied after any surface skeletonization algorithm and results in a curve skeleton. Moreover, the classification that we use can deal also with two-voxel thick surfaces so that our curve skeletonization algorithm can also be applied after algorithms resulting in two-voxel thick surface skeletons.

2 Notions

We refer to bi-level images consisting of object and background. In particular, in this paper the object is a surface, e.g., the one resulting after a 3D solid object has been reduced to its surface skeleton. The 26-connectedness is chosen for the object and the 6-connectedness for the background. Any voxel v has three types of neighbours: face, edge, and point neighbours.

We will use two different surface skeletonization algorithms for the examples in this paper. One is based on the D^6 metric, i.e., the 3D equivalent of the city-block metric, and was introduced in [9]. We will call the resulting set D^6 *surface skeleton*. The other algorithm is based on the D^{26} metric, i.e., the 3D equivalent of the chess board metric, and was introduced in [11]. We will call the resulting set D^{26} *surface skeleton*.

Classification of the voxels in a surface was suggested in [3,6]. A voxel is classified after investigating its $3 \times 3 \times 3$ neighbourhood. Of course, that classification works for an "ideal" surface, i.e., a surface which is one-voxel thick everywhere. Complex cases consisting of surfaces crossing each other would not produce a consistent classification at junctions, whenever these are more than one-voxel thick, see Fig.3. In Fig.3, left, voxels where the two surfaces cross each other, shown in dark grey, are classified as junction voxels, while in Fig.3, right, voxels where the two surfaces cross, marked by •, are classified as inner voxels. That classification also fails when applied to surface skeletons of 3D objects having regions whose thickness is an even number of voxels. These surface skeletons are in fact likely to be two-voxel thick, [9,11].

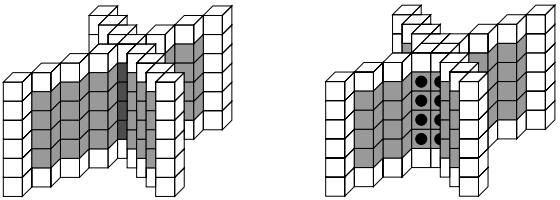

Fig. 3. Simple examples of junctions between surfaces. Edge voxels are shown in white, inner voxels in grey, and junction voxels in dark grey for the classification introduced in [3,6]. Voxels marked by • should be classified as junction voxels to be consistent.

In this paper, we use the classification introduced and thoroughly described in [8]. There some criteria suggested in [3,6] are used in combination with other criteria, where a slightly larger neighbourhood of each voxel is taken into account. Two-voxel thick regions are singled out with a linear four-voxel configuration ($4 \times 1 \times 1$, $1 \times 4 \times 1$, $1 \times 1 \times 4$), which identifies portions of the surface skeletons being exactly two-voxel thick in any of the x, y, z-directions. The classification requires a number of different criteria and the same voxels are likely to be checked against many criteria before they are eventually classified. It is then not possible to summarize it here. A more detailed description is given in a recently submitted paper [4].

3 Curve Skeletonization by Junction Detection

The different classes of voxels in the surface skeleton we are interested in are *junction*, *inner*, *edge*, and *curve* voxels, see Fig.1. The curve skeleton is obtained in two steps by an iterative algorithm. Each iteration of both steps includes two subiterations dealing with i) detection of edge voxels and ii) voxel removal by means of topology preserving removal operations, respectively. The two steps differ from each other for the selection of the voxels that, at each iteration, are checked to identify the edge voxels, i.e., the set of voxels candidate for removal.

In the first step, only voxels *initially* (i.e., on the original surface skeleton) classified as inner (or edge) voxels are checked during the identification of the edge voxels. Voxels are actually interpreted as edge voxels, if their neighbourhood has been suitably modified due to removal of some neighbouring voxels. Note that 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. An undesirable branch shortening would be obtained if also the voxels initially classified as junction voxels were checked. In fact, voxels placed on the tips of junctions, i.e., the junction voxels that should play the role of end-points, 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 during the identification of the edge voxels. Edge voxels that have been transformed into curves during the first step are not interpreted as edge voxels and, hence, are automatically preserved from removal. The remaining voxels initially classified as junction voxels can be now interpreted as edge voxels, if their neighbourhood has been suitably modified.

In both steps, on the current set of edge voxels, removal is done unless voxels are necessary for topology preservation. Standard topology preserving removal operations, e.g., those described in [11], are sequentially applied. After each subiteration of removal of edge voxels, a new iteration starts and a new set of edge voxels is determined. Removal operations are then applied on the new set of edge voxels. Edge detection and voxel removal are iterated until no more edge voxels are identified and possibly removed.

If the set of junctions of the surface skeleton has only peripheral junctions, i.e., no junctions are grouped into loops, the curve skeleton is obtained directly after the first step. It consists of the initial curve and junction voxels, as well as voxels necessary for connectedness. Otherwise, also the second step is necessary. In this case, the effect of the first step is to cause some junctions initially forming loops (see Fig.2) to become edge voxels. This allows skeletonization to continue towards voxels in the innermost part of the surface skeleton.

Our algorithm is first illustrated on the D^6 surface skeleton of a simple object, a cube, for which the first step is enough to compute the curve skeleton, Fig.4. The D^6 surface skeleton of the cube is shown in the middle. The resulting curve skeleton, shown to the right, coincides with the set of junction voxels.

A slightly more complex case, the D^6 surface skeleton of a box, is shown in Fig.5(b). The set resulting at completion of the first step of our algorithm is shown in Fig.5(c). Voxels detected as junction voxels during the initial clas-

Fig. 4. A cube with its D^6 surface skeleton and the curve skeleton computed by our algorithm.

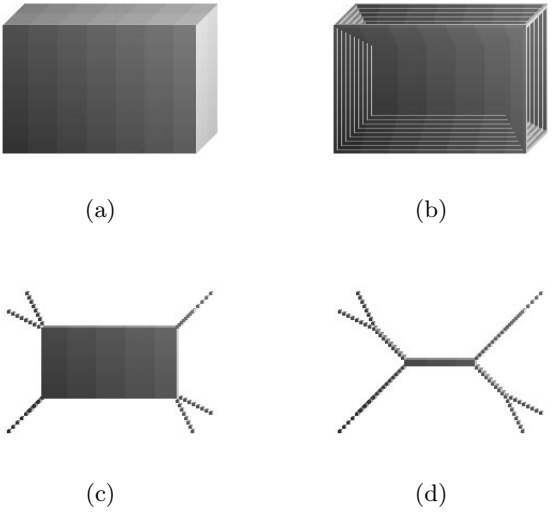

Fig. 5. A box with its D^6 surface skeleton, top. The intermediate result and the final result of the curve skeletonization algorithm, bottom.

sification are partly transformed into curve voxels, and partly into edge voxels surrounding the rectangular surface found in the middle of the box. The curve skeleton is the set resulting after the second step of our algorithm, see Fig.5(d).

The box in Fig.5(a) is of size $60 \times 40 \times 20$ voxels, i.e., it has an even number of voxels in every direction. The rectangular surface in the middle of the surface skeleton is hence two-voxel thick. Therefore, also the obtained curve skeleton is two-voxel thick in the central part. Final reduction to a one-voxel thick curve skeleton could be achieved by identifying tip of protrusions and iteratively removing voxels not necessary for topology preservation as was shown in [2].

For the sake of completeness, we point out that the surface skeleton 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 (the rectangular surface can be one-voxel thick in depth, but an even number of voxels in other directions, as in the case above). Also, we have found that if reduction to one-voxel thickness is postponed until the curve skeleton has been obtained, the risk of creating spurious branches in the curve skeleton is significantly reduced.

4 Some Examples

Projections of thin complex structures are hard to visualize in a descriptive way. We are showing the results of our algorithm on rather small synthetic objects. In Figs. 6 and 7, a pyramid rotated 45° with its D^6 and D^{26} surface skeletons and the curve skeletons computed by our algorithm are shown. In Figs. 8 and 9, a cylinder with its D^6 and D^{26} surface skeletons and the curve skeletons computed by our algorithm are shown.

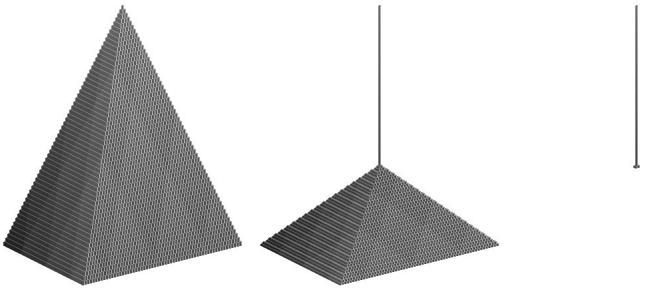

Fig. 6. A pyramid rotated 45° with its D^6 surface skeleton and the curve skeleton computed by our algorithm.

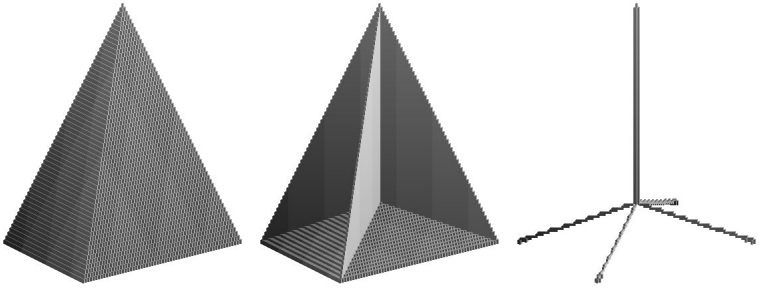

Fig. 7. The same pyramid as in Fig.6 with its D^{26} surface skeleton and the curve skeleton computed by our algorithm.

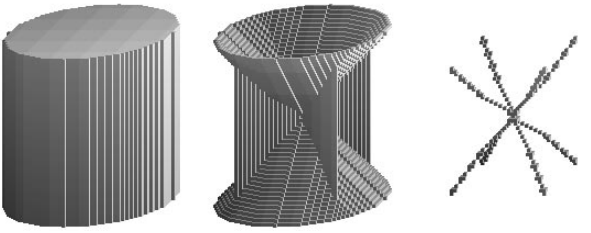

Fig. 8. A cylinder with its D^6 surface skeleton and the curve skeleton computed by our algorithm.

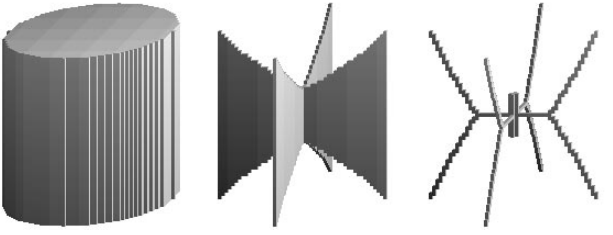

Fig. 9. The same cylinder as in Fig.8 with its D^{26} surface skeleton and the curve skeleton computed by our algorithm.

We remark that the curve skeletonization algorithm can be applied regardless of which algorithm has been used to compute the surface skeleton. In any case, the curve skeleton is a satisfactory shape descriptor.

The curve skeleton of the D^{26} surface skeleton of the cylinder, Fig.9, right, has a number of peripheral branches besides those including voxels initially classified as junction voxels. This is due to the fact that the edges of the surfaces are characterized by convexities (angle less than 90°), which during iterated voxel removal are shrunk to curves. These curves are very short as they consist of one or two voxels only. However, once voxels have been classified as curve voxels they are ascribed to the skeleton and this causes further voxels to be prevented from removal during curve skeletonization.

5 Conclusion

In this paper, we have presented an algorithm that computes the curve skeleton of a 3D solid object starting from its surface skeleton. The algorithm is based on the detection of curve and junction voxels in the surface skeleton. One of

the advantages of this approach is its independence of the choice of surface skeletonization algorithm. This is not always the case with other algorithms. For example, the surface skeleton → curve skeleton part of the algorithm presented in [2] can only be computed when starting from a D^6 surface skeleton, to obtain a reasonable result. In fact, it includes a blind end-point detection criterion tailored specifically to the D^6 case, which would not work nicely in other cases, e.g., for a D^{26} surface skeleton.

The computational cost of the curve skeletonization algorithm is quite high (a couple of minutes for complex real objects in images of size $128 \times 128 \times 128$ voxels). This is due to the non-optimized classification process that has to be repeatedly used during curve skeletonization.

We have tested our algorithm on a large number of surface skeletons, one-voxel and two-voxel thick, and have in all cases obtained satisfactory results.

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

References

1. G. Bertrand and Z. Aktouf. A three-dimensional thinning algorithm using subfields. In R. A. Melter and A. Y. Wu, editors, *Vision Geometry III*, pages 113–124. Proc. SPIE 2356, 1994.
2. G. Borgefors, I. Nyström, and G. Sanniti di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.
3. G. Malandain, G. Bertrand, and N. Ayache. Topological segmentation of discrete surfaces. *International Journal of Computer Vision*, 10(2):183–197, 1993.
4. I. Nyström, G. Sanniti di Baja, and S. Svensson. Curve skeletonization guided by surface voxel classification. Submitted to *Pattern Recognition Letters*, 2001.
5. K. Palágyi and A. Kuba. A parallel 3D 12-subiteration thinning algorithm. *Graphical Models and Image Processing*, 61:199–221, 1999.
6. P. K. Saha and B. B. Chaudhuri. Detection of 3-D simple points for topology preserving transformations with application to thinning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1028–1032, Oct. 1994.
7. P. K. Saha, B. B. Chaudhuri, and D. D. Majumder. A new shape preserving parallel thinning algorithm for 3D digital images. *Pattern Recognition*, 30(12):1939–1955, Dec. 1997.
8. G. Sanniti di Baja and S. Svensson. Classification of two-voxel thick surfaces: a first approach. Internal Report 19, Centre for Image Analysis, 2000. Available from the authors.
9. G. Sanniti di Baja and S. Svensson. Surface skeletons detected on the D^6 distance transform. In F. J. Ferri, J. M. Iñetsa, A. Amin, and P. Pudil, editors, *Proceedings of S+SSPR 2000: Advances in Pattern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 387–396, Alicante, Spain, 2000. Springer-Verlag, Berlin Heidelberg.
10. S. N. Srihari, J. K. Udupa, and M.-M. Yau. Understanding the bin of parts. In *Proceedings of International Conference on Cybernetics and Society, Denver, Colorado*, pages 44–49, Oct. 1979.

11. S. Svensson, I. Nyström, and G. Borgefors. Fully reversible skeletonization for volume images based on anchor-points from the D^{26} distance transform. In B. K. Ersbøll and P. Johansen, editors, *Proceedings of The 11th Scandinavian Conference on Image Analysis (SCIA'99)*, pages 601–608, Kangerlussuaq, Greenland, 1999. The Pattern Recognition Society of Denmark.
12. Y.-F. Tsao and K.-S. Fu. Parallel thinning algorithm for 3-D pictures. *Computer Graphics and Image Processing*, 17(4):315–331, Dec. 1981.