Surface Skeletons Detected on the D^6 Distance Transform

Gabriella Sanniti di Baja¹ and Stina Svensson²

¹ Istituto di Cibernetica, Italian National Research Council
Via Toiano 6, IT-80072 Arco Felice (Naples), ITALY
`gsdb@imagm.cib.na.cnr.it`

² Centre for Image Analysis, Swedish University of Agricultural Sciences
Lägerhyddvägen 17, SE-75237 Uppsala, SWEDEN
`stina@cb.uu.se`

Abstract. We present an algorithm for extracting the surface skeleton of a 3D object from its D^6 distance transform. The skeletal voxels are directly detected and marked on the distance transform within a small number of inspections, independent of object thickness. This makes the algorithm preferable with respect to algorithms based on iterative application of topology preserving removal operations, when working with thick objects. The set of skeletal voxels is centred within the object, symmetric, and topologically correct. It is at most 2-voxel wide (except for some cases of surface intersections) and includes all centres of maximal D^6 balls, which makes skeletonization reversible. Reduction to a unit wide surface skeleton can be obtained by suitable post-processing.

1 Introduction

The skeleton of a digital object is a convenient tool for shape analysis. A commonly followed approach to skeletonization is based on the use of topology preserving removal operations. To have a skeleton centred within the object and hence reflecting its geometrical features, removal operations have to be repeatedly applied border after border. This implies generally long computation time, proportional to object thickness, which prevents the actual use of the skeleton for real time applications. An alternative approach, proposed only for 2D objects, is based on the use of the distance transform. In [1], the city-block distance transform of a 2D object is considered. On the distance transform, the layers, i.e., the sets of pixels having the same distance label, can be interpreted as the successive borders that would characterize the object when this undergoes iterated pixel removal. On these layers the skeletal pixels can be identified and marked in a small and fixed number of inspections. In this paper, we present an algorithm to compute the surface skeleton of a 3D object which follows the marking procedure introduced in [1] and is based on the skeletonization operations introduced in [4,6]. The obtained surface skeleton is at most 2-voxel wide (except for some cases of surface intersections) and includes all the centres of maximal balls. This makes our procedure reversible. Reduction to a unit wide surface skeleton can be

obtained, e.g., by using the process described in [4]. In this paper, however, we favour full reversibility and do not perform final thinning. Our surface skeleton is centred within the object and is symmetric. It is obtained in a small number of inspections, which makes its use convenient for applications when thick objects have to be skeletonized.

2 Definitions and Notions

The images considered here are volume images consisting of object and background, i.e., 3D bi-level images. The central voxel in a $3 \times 3 \times 3$ neighbourhood has three types of neighbours: face, edge, and point neighbours. In Fig. 1 the $3 \times 3 \times 3$ neighbourhood of a voxel (white) is shown. The six face neighbours, the twelve edge neighbours, and the eight point neighbours are shown in light grey, grey, and dark grey, respectively. 26-connectedness is used for the object and 6-connectedness is used for the background.

Fig. 1. Numbering of the voxels in a $3 \times 3 \times 3$ neighbourhood of a voxel v , denoted by 13 and shown in white. Face neighbours are shown in light grey, edge neighbours in grey and point neighbours in dark grey.

An object has a *tunnel* if there exists a closed 26-connected path in the object that can not be deformed to a single voxel (for details, see [7]), and a *cavity* if a background component is fully enclosed in the object.

The surface skeleton is a set of voxels with the same number of components, cavities and tunnels as the object; it has unit thickness and is centred within the object. Skeletonization can be performed by iteratively applying topology preserving removal operations to the object, border after border. A number of iterations proportional to the maximal object thickness is necessary. Such a skeleton is generally not symmetric, because only voxels indispensable for object connectedness are kept, and identification of the skeletal voxels depends on the order in which border voxels are examined. This is a drawback of iterative skeletonization, since symmetry is an important feature for shape analysis.

A voxel v can safely be removed if its removal does not change the number of object components, cavities, or tunnels in its $3 \times 3 \times 3$ neighbourhood, [2, 8]. To verify whether the number of object components changes, the number of 26-connected components (N^{26}) in the 26-neighbourhood of v can be counted.

Fig. 2. A D^6 ball with radius 15.

Object voxels with $N^{26} \neq 1$ should be ascribed to the skeleton. If $N^{26} > 1$, removal of v would break the connectivity, i.e., a component would break into two or more components. If $N^{26} < 1$, removal of v would cause vanishing of an object component consisting of one voxel. To preserve the number of cavities and tunnels, the number of 6-connected background components in the 18-neighbourhood that are 6-adjacent to v ($\overline{N}_f^{18}$) is counted. Voxels with $\overline{N}_f^{18} \neq 1$ should be ascribed to the skeleton. In fact, if $\overline{N}_f^{18} < 1$ or $\overline{N}_f^{18} > 1$ removal of v would create a cavity or a tunnel, respectively.

In the *distance transform* (DT), each object voxel is labelled by the distance to the closest background voxel, [3]. In this paper we will use the D^6 distance, i.e., the number of steps in a minimal 6-connected path between voxels, to compute the distance transform (DT^6). This is the 3D equivalent of the city-block distance transform. The distance value of a voxel of DT^6 can be interpreted as the radius of a D^6 ball centred on the voxel and included in the object. (Note that the shape of a D^6 ball is an octahedron, see Fig. 2.) An object voxel in DT^6 is a *centre of maximal ball* (CMB) if the D^6 ball centred on the voxel is not completely covered by any other single ball in the object. On DT^6 , a voxel is a CMB if none of its face neighbours have a higher distance value. From the set of CMBs the whole object can be recovered by applying the *reverse distance transformation*.

Our skeletonization algorithm does not require the explicit computation of N^{26} , since the object connectedness is preserved by other conditions. A voxel satisfying any of the following conditions, introduced in [4], is marked as a skeletal voxel:

Condition A1: No pair of opposite face neighbours of a voxel v exists such that one is a background voxel and the other is an internal voxel. (See Fig. 3, left.)

Condition A2: There exists an edge neighbour, e , of v which is a border voxel, such that the two voxels that are face neighbours of both e and v are background voxels. (See Fig. 3, middle.)

Condition A3: There exists a point neighbour, p , of v which is a border voxel, such that the six voxels that are neighbours of both p and v are background voxels. (See Fig. 3, right.)

To avoid changing the number of cavities and tunnels, the condition we use is based on the computation of $\overline{N}_f^{18}$, performed by means of the computationally convenient method introduced in [5].

Fig. 3. Voxels involved in *Condition A1*, left, *Condition A2*, middle, and *Condition A3*, right. White, grey, and black denote background voxels, border voxels, and internal voxels, respectively.

In the following, we use a digital Euclidean ball with radius 28, shown in Fig. 4, left, as a running example. This is a difficult object because the skeleton that we compute is based on DT^6 . The set of CMBs of the object is shown to the right of the ball. The shape of the resulting skeleton should be determined by the CMBs.

3 Marking the Skeletal Voxels

We distinguish three types of skeletal voxels, *intrinsic* skeletal voxels, *induced* skeletal voxels, for short intrinsic voxels and induced voxels, and *tunnel* voxels. Intrinsic voxels and induced voxels are necessary to guarantee object connectedness and to prevent changing the number of cavities. Tunnel voxels are necessary to prevent changing the number of tunnels. Skeletonization is done in 3 steps. Intrinsic voxels are found during the first step. Induced voxels are found during the second and the third steps. Tunnel voxels are found during the third step.

Once DT^6 has been computed, intrinsic voxels can be identified in a parallel fashion, while induced voxels can be found only after their neighbourhood has been modified by the detection of other skeletal voxels. Different interpretations of neighbouring voxels are necessary for intrinsic and induced voxels. For intrinsic voxel detection, neighbours of a voxel v with label smaller than v are interpreted as belonging to the background, neighbours with the same label as v are seen as border voxels, and neighbours with label larger than v are interpreted as internal voxels. For induced voxel detection, the presence of marked voxels (i.e., skeletal voxels) in the neighbourhood has to be taken into account. To be specific, marked neighbours with label smaller than v are interpreted as border voxels. For all other neighbours, including the remaining marked voxels, the interpretation used for intrinsic voxel detection is followed.

Intrinsic voxels include all CMBs as well as voxels that are not CMBs, but are placed in saddle configurations, *saddle voxels*. Intrinsic voxels could all be detected by using *Condition A1*, but we specifically detect the CMBs to distinguish them from the saddle voxels. We identify and mark parallelwise all intrinsic voxels during one scan of DT^6 . (First step of the skeletal voxel detection process.)

Induced voxels are identified in two steps depending on whether they are induced by the CMBs, or are induced by other skeletal voxels (saddle voxels or other already induced voxels). The voxels induced by the CMBs have label equal

to or larger than the label of the inducing CMBs. They are found and marked during one scan of DT^6 . (Second step of the detection process.) The remaining induced voxels have label larger than the label of the inducing voxels. They are found and marked during the third step, which consists of eight scans of DT^6 , possibly repeated. In general, one set of eight scans is enough to mark all the induced voxels, even for complex and thick objects. Note that no more than two sets of eight scans are necessary, due to the constraints posed by the distance transformation. During the second and third steps *Conditions A1, A2, and A3* are used.

Since voxels that would be classified as internal voxels by iterative skeletonization are directly detected by *Condition A1*, there is no need of an extra condition for preventing creation of cavities. Moreover, since marking a voxel by means of *Conditions A1, A2, and A3* does not change the number of object components in the $3 \times 3 \times 3$ neighbourhood of that voxel, no merging of cavities can occur. Hence, only tunnel creation has to be prevented. Conditions based on the computation of $\overline{N}_f^{18}$ are used to find tunnel voxels during the third step. To favour maintenance of symmetry, tunnel voxels are detected only starting from the fifth scan, since most of the induced voxels have already been found by then. In general after one complete set of eight scans, all tunnel voxels are found. However, for complex objects, repetitions of the eight scans could be necessary.

Since detection of the intrinsic voxels can be performed straight forwardly, no further details are provided here. Description of induced and tunnel voxel detection is given below.

In the following sections, both the voxel and its associated distance value will be denoted by the same letter.

3.1 Induced Voxel Detection

During the second step, for each CMB, say s , the $3 \times 3 \times 3$ neighbourhood of s is inspected. The maximum distance label, max , is computed within each 26-connected component of neighbours of s with label equal to or larger than s . For each voxel v in the $3 \times 3 \times 3$ neighbourhood of s , such that $v = max$, *Conditions A1, A2, and A3* are checked for v , which is possibly marked. Marking is done in a parallel way during this step, which consists of one scan.

With reference to the following $3 \times 3 \times 3$ neighbourhood

0 0 0	0 1 1	1 1 2
0 1 1	1 2 2	2 2 <u>3</u>
1 1 2	2 2 <u>3</u>	3 <u>3</u> <u>4</u>

where the central CMB labelled 2 is shown in bold, the only voxel v for which *Conditions A1, A2, and A3* can be checked is the underlined 4, since 4 is the maximum label in the neighbourhood. The 4 is actually marked due to *Condition A3*. Checking only voxels $v = max$, instead of all voxels $v \geq s$, is done to limit the number of induced voxels. If all voxels $v \geq s$ were checked, also the italic 3's were marked (in parallel) by *Condition A2*. These voxels are superfluous for

preservation of object connectedness, hence their detection would only produce unwanted thickening of the set of the skeletal voxels.

With reference to the small part of DT^6 below

0 0 0 0	0 0 0 0	0 0 0 0
0 0 0 0	0 1 0 1	0 0 0 0
0 0 0 1	0 0 <u>1</u> 2	0 0 0 1
0 0 1 2	0 1 2 3	0 0 1 2

we note that the maximum distance label in the neighbourhood of the bold CMB labelled 1 is 1. If the underlined 1 is not allowed to be checked against *Conditions A1, A2, and A3*, then skeleton connectedness could not be achieved. For this reason, when we compute the maximum distance label of the neighbours of s , we include also neighbours labelled as s . (Note that in the neighbourhood of any other voxel s , that is not CMB, the maximum distance label is always larger than the label of s .)

During the second step some of the saddle voxels can be removed, if superfluous for connectedness preservation. This is the case when a saddle voxel is neighbour of both a CMB and the voxel(s) induced by that CMB in step 2. This can be seen with reference to the small part of DT^6 below

0 0 0 0	1 1 0 0	<u>2</u> 1 0 0
0 1 0 0	<i>1</i> 2 1 0	1 1 0 0
0 0 0 0	0 1 0 0	0 0 0 0
0 0 0 0	0 0 0 0	0 0 0 0

showing a CMB (label 2, bold), and two saddle voxels (labelled 1, italic). The CMB induces, by *Condition A3*, marking of a voxel (label 2, underlined), which makes the two saddle voxels superfluous. Thus we remove their markers.

For our running example, the result after the second step is shown in Fig. 4, right, and to the left of this, the voxels added by step 2.

Fig. 4. From left to right, a digital Euclidean ball with radius 28, the set of CMBs, the voxels added by step 2, and the result of steps 1 and 2.

The remaining induced voxels are detected during the third step, which is performed by repeatedly inspecting DT^6 . Any voxel v , neighbour of an already

marked voxel s , such that $v > s$ and such that v is the maximum distance label in the $3 \times 3 \times 3$ neighbourhood of s is checked against *Conditions A1*, *A2*, and *A3*. To keep the symmetry of the skeleton, *Conditions A1*, *A2*, and *A3* are only checked in the direction of the already visited neighbours of v , including the inducing voxel s . When a voxel is inspected, one of its point neighbours, three of its edge neighbours and three of its face neighbours are already visited. This implies that for each inspection, *Condition A3* is considered in one direction out of eight possible ones, *Condition A2* in three directions out of twelve possible ones, and *Condition A1* in three directions out of six possible ones. We need eight inspections, so that *Condition A3* is checked in all directions. These eight inspections can be arranged in such a way that after four inspections *Condition A2* and *A1* are checked in all possible directions. We propose the following inspection order, where, for the neighbours, the numbering shown in Fig. 1 is used, see Table 1. The order implies that after four inspections, *Condition A2* is investigated once in every direction and *Condition A1* twice. The inspections are repeated as far as markers are set. Different ordering of the inspections will produce the same result, but the number of inspections could be larger.

Table 1. Proposed inspection order for the third step.

x	y	z	point	edge			face		
$\rightarrow$	$\downarrow$	$\swarrow$	0	1	3	9	4	10	12
$\leftarrow$	$\uparrow$	$\swarrow$	8	5	7	17	4	14	16
$\leftarrow$	$\downarrow$	$\nearrow$	20	11	19	23	10	14	22
$\rightarrow$	$\uparrow$	$\nearrow$	24	15	21	25	12	16	22
$\leftarrow$	$\downarrow$	$\swarrow$	2	1	5	11	4	10	14
$\rightarrow$	$\uparrow$	$\swarrow$	6	3	7	15	4	12	16
$\rightarrow$	$\downarrow$	$\nearrow$	18	5	7	17	10	12	22
$\leftarrow$	$\uparrow$	$\nearrow$	26	17	23	19	14	16	22

3.2 Tunnel Voxel Detection

To avoid creation of tunnels, voxels with $\overline{N}_f^{18} > 1$ are marked. However, this is not always enough. With reference to the part of DT^6 shown below

0 0 0 0 0 0 0	0 0 0 0 0 1 1	0 0 0 0 1 1 1
0 0 0 0 0 1 1	0 0 0 1 1 2 2	0 0 0 1 2 2 2
0 0 0 1 1 2 2	0 0 1 2 <u>2</u> 2 2	0 0 1 2 3 2 2
0 0 1 2 2 1 1	0 1 2 3 <u>2</u> 1 1	0 1 2 3 2 1 1
0 0 1 2 1 0 0	0 1 <u>2</u> 2 1 0 0	1 2 3 2 1 0 0
0 1 2 1 0 0 0	1 2 2 1 0 0 0	1 2 2 1 0 0 0
0 1 2 1 0 0 0	1 2 2 1 0 0 0	1 2 2 1 0 0 0

where voxels marked during the first two steps are shown in bold, we see that at least one of the 2's, in italic, and one of the 2's, underlined, have to be

marked to avoid creation of two tunnels. To have a symmetric skeleton, all four 2's should indeed be marked. But for all these voxels, $\overline{N}_f^{18} = 1$. Note that if an iterative skeletonization algorithm is used, the condition based on $\overline{N}_f^{18}$ is enough to prevent tunnel creation, but an asymmetric skeleton is obtained. In fact, depending on the order in which border voxels are examined on the border "2", for only one 2 in each of the two pairs (*italic and underlined*) is $\overline{N}_f^{18} = 2$.

We need a special condition to fill tunnels consisting of pairs of equilabelled voxels. If voxels with the same distance value as the central voxel are interpreted as belonging to the background (instead of to the border!) and $\overline{N}_f^{18}$ is computed in a parallel fashion with this interpretation, then $\overline{N}_f^{18} = 2$ for the all four 2's. To avoid ambiguities, we will denote by $\hat{N}_f^{18}$ the number of 6-connected background components 6-adjacent to the central voxel in the 18-neighbourhood, when the neighbours labelled as the central voxel are interpreted as belonging to the background.

Indeed tunnel voxels should be identified only after tunnels are created. Thus, tunnels should be filled only after all voxels detected by *Conditions A1, A2, and A3* are marked. To fill the tunnels symmetrically, an iterative procedure should be used, active layer after layer in parallel. This requires a number of iterations dependent on the size of the tunnels to be filled. Since this can be rather large, e.g., when a tunnel is due to two diverging curves originating from the same CMB, an iterative method is not convenient. An alternative would be that of filling the tunnels, still after all voxels detected by *Conditions A1, A2, and A3* have been marked, by repeated sequential inspections of DT^6 . This might reduce the computation time, but tunnel voxels could be marked asymmetrically due to the sequentiality of the process. A compromise is necessary between the desires of having a symmetric skeleton and of reducing the computation time. To speed up the process, we perform tunnel detection sequentially, and include this detection in the third step. (Note that tunnel voxels do not induce any voxel to be marked by *Conditions A1, A2, and A3*.) To limit the asymmetries, we start tunnel detection only from the fifth scan, since most of the skeletal voxels have already been marked by then. We are aware that the scans needed to detect the induced voxels might not be sufficient to fill all tunnels, if these have complex structures. Thus the third step, which includes tunnel filling might need to be repeated more than twice to fill all tunnels.

Besides by starting tunnel filling from the fifth scan, we reduce the possibility of creating asymmetries also in the following way. During each scan, marking by $\overline{N}_f^{18} > 1$ is done sequentially and marking by $\hat{N}_f^{18} > 1$ is done parallelwise. $\overline{N}_f^{18}$ is computed by ignoring markers set by $\hat{N}_f^{18}$ during the same scan, and vice versa. Some voxels, for which $\hat{N}_f^{18} > 1$, are redundant if a face neighbour with the same distance value is marked by $\overline{N}_f^{18} > 1$ during the same scan. Therefore, if the currently visited voxel v is marked by $\overline{N}_f^{18} > 1$, any already visited face neighbour with the same label and marked by $\hat{N}_f^{18} > 1$, has the marker removed;

moreover, any not yet visited face neighbour with the same label is prevented from being marked by $\hat{N}_f^{18} > 1$ during the same scan.

For our running example, the resulting skeleton is shown in Fig. 5, right. The result of steps 1 and 2 and the voxels added by the third step are shown to the left and in the middle, respectively. Two other examples are shown in Fig. 6. All images used in this paper are $64 \times 64 \times 64$ voxels. The number of scans of DT^6 necessary to find all skeletal voxels, including tunnel voxels, is 8 (1+1+6) for the running example, and 9 and 8 for the two examples in Fig. 6. Even if it might be difficult to see from the Figs., the three surface skeletons *are* symmetric. Of course they are also centred within the objects, includes all CMBs and are at most 2-voxels thick.

Fig. 5. The result of steps 1 and 2, left, the voxels added by step 3, middle, and the resulting skeleton, right.

Fig. 6. Two objects and their surface skeletons.

4 Conclusion

A skeletonization algorithm based on the inspection of the D^6 distance transform was presented. The resulting surface skeletons are symmetric, fully reversible, centred in the object (with respect to D^6), topologically correct, and at most 2-voxel wide. To obtain unit-wide skeletons, post-processing could be done (not performed in this paper). Since the skeleton is based on the inspection of the distance transform, the number of scans does not depend on the thickness of the object (even if tunnel filling depends on object complexity and we can not claim that our skeletonization ends in an a priori known number of scans). Thus, the computation time is smaller than the time required by iterative skeletonization, when thick objects are considered.

Further work is in progress to improve tunnel filling and to extend the algorithm to other distance transforms.

Acknowledgement

Prof. Gunilla Borgefors and Dr. Ingela Nyström, both Centre for Image Analysis, Uppsala, Sweden, are gratefully acknowledged for their valuable support during the development of the algorithm described here.

This work was partially supported by a grant to Svensson from International Office at Swedish University of Agricultural Sciences.

References

- [1] C. Arcelli and G. Sanniti di Baja. A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):411–414, Apr. 1989.
- [2] G. Bertrand and G. Malandain. A new characterization of three-dimensional simple points. *Pattern Recognition Letters*, 15:169–175, Feb. 1994.
- [3] G. Borgefors. On digital distance transform in three dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, 1996.
- [4] G. Borgefors, I. Nyström, and G. Sanniti di Baja. Surface skeletonization of volume objects. In P. Perner, P. Wang, and azriel Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition (SSPR'96)*, pages 251–259, Leipzig, Germany, Aug. 1996. Springer-Verlag. LNCS 1121.
- [5] G. Borgefors, I. Nyström, and G. Sanniti di Baja. Connected components in 3D neighbourhoods. In M. Frydrych, J. Parkkinen, and A. Visa, editors, *Proceedings of 10th Scandinavian Conference on Image Analysis (SCIA'97)*, pages 567–572, Lappeenranta, Finland, 1997. Pattern Recognition Society of Finland.
- [6] G. Borgefors, I. Nyström, and G. Sanniti di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, July 1999.
- [7] T. Y. Kong. A digital fundamental group. *Computers & Graphics*, 13(2):159–166, 1989.
- [8] P. K. Saha and B. B. Chaudhuri. 3D digital topology under binary transformation with applications. *Computer Vision and Image Understanding*, 63(3):418–429, May 1996.