Skeletonization of Digital Objects

Gabriella Sanniti di Baja

Istituto di Cibernetica "E. Caianiello" – CNR, Pozzuoli (Naples), Italy
g.sannitidibaja@cib.na.cn.it

Abstract. Skeletonization is a way to reduce dimensionality of digital objects and is of interest in a number of tasks for image analysis. In this paper, efficient approaches to skeletonization of 2D and 3D digital objects are illustrated.

1 Introduction

Skeletonization is a process leading to the extraction of a medial representation of a digital object with lower dimension. Algorithms to compute such a medial representation have been called, besides skeletonization, also thinning and medial axis transformation, and the resulting set has received names such as medial line, medial axis transform, skeleton, and labeled skeleton. These medial representations do not necessarily coincide and are characterized by different properties. In this paper, we refer to skeletonization as to a process to compute a subset of the object, the skeleton, topologically homotopic to the object, spatially placed along the medial region of the object and, depending on the problem domain, expected to account for different geometrical and morphological properties of the represented object. In particular, all object parts regarded as significant in the problem domain should be reconstructed starting from the skeleton (i.e., the skeleton enjoys the reversibility feature).

Skeletonization has been initially suggested to compute linear representations of 2D digital objects. During the process, object pixels on the border of the object are changed to background pixels until a subset S of the object is obtained, which is a union of arcs and curves placed symmetrically with respect to the border of the object.

The research on 2D skeletonization has been largely influenced by the work of Blum dealing with a geometry based on the primitive notions of a symmetric point and a growth process [1,2]. A point p of a continuous object at distance d from the boundary of the object is symmetric if at least two points of the boundary exist, having distance d from p. A maximal ball can be associated to every symmetric point p by growth of the symmetric point, where the maximal ball is the largest ball centered on p and contained in the object. The set of symmetric points, labeled with the radii of the associated maximal balls, constitutes the medial axis transform of the object. The union of the maximal balls coincides in shape and size with the object, thus the medial axis transform enjoys the reversibility feature.

The medial axis transform is not necessarily a minimal coding of the object, since a maximal ball can be completely included by the union of other maximal balls. An object can often be reconstructed from proper subsets of its medial axis transform. Moreover, the medial axis does not necessarily reflect the topological properties of the object. For example, if a connected object consists of a number of balls tangent to

J.F. Martínez-Trinidad et al. (Eds.): CIARP 2006, LNCS 4225, pp. 1 – 13, 2006.

each other, its medial axis is the non-connected set consisting of the centers of the balls. Differently from the medial axis transform, the skeleton is a subset of the object, which includes, besides the centers of maximal balls (so that the skeleton enjoys the reversibility feature), also other elements necessary to guarantee that the object and its skeleton are homotopic.

Reducing digital objects to lower dimensions is even more desirable in 3D. The 3D skeleton is either a set of 3D surfaces and curves (called surface skeleton, *SS*) or, if the object does not include cavities, it may consist only of 3D curves (called curve skeleton, *CS*). The basic idea underlying 3D skeletonization does not differ from the 2D case: object voxels are changed to background voxels, provided that topology and geometry are not altered, until the skeleton is obtained.

The literature in the field of skeletonization is very rich and different approaches can be followed to compute the skeleton. A well-known approach is based on the repeated use of topology preserving removal operations. These are applied border after border so as to have a skeleton centered within the object, and hence reflecting its geometrical features. A drawback of this approach, when the algorithm is implemented on conventional sequential computers, is that a number of image scans proportional to the object's thickness is necessary to reach the goal, so that the computation time is quite large. A computationally convenient approach is based on the use of the distance transform of the object. In the distance transform, the layers (i.e., the sets of object elements at the same distance from the background) can be interpreted as the successive borders that characterize the object when this undergoes skeletonization by iterated removal of border elements. In principle, since all layers are available in the distance transform, the skeletal elements can be directly identified and marked in a small and fixed number of inspections of the distance transform.

Both continuous and discrete approaches have been suggested in the literature. We will here focus on discrete approaches to skeletonization. A brief overview of different approaches can be found in [3,4]. This paper is organized as follows. In Section 2, preliminary notions are given. Sections 3 and 4 respectively deal with 2D and 3D skeletonization. Some concluding remarks are given in Section 5.

2 Notions and Definitions

We work with a binary discrete image I in square/cubic grid, where the object O is the set of 1's and the background B is the set of 0's. When working in the discrete space, some problems specific to this space and relevant for skeletonization have to be faced. It is well known that a different connectivity type has to be used for the object and for its complement (the background) to avoid topological paradoxa. The connectivity type depends on which, among the neighbors of a pixel/voxel, are considered as directly connected to the pixel/voxel at hand.

In 2D, a pixel p has four neighbors sharing an edge with p, and four neighbors sharing a vertex with p. The 4-connectivity considers as connected to p the pixels sharing an edge with p, while the 8-connectivity considers both kinds of neighbors.

In 3D, a voxel v has six neighbors sharing a face with v, twelve neighbors sharing an edge and eight neighbors sharing a vertex. Three connectivity types are hence possible: 26-connectivity, when all three kinds of neighbors are considered,

18-connectivity, when neighbors sharing a face or an edge are considered, and 6-connectivity, when only the neighbors sharing a face are considered.

If the same connectivity type is used for both O and B, a closed curve/surface would not divide its complement into disjoint parts, or an open curve/surface would divide its complement into disjoint parts. In the 2D discrete space, the 8-connectivity and the 4-connectivity are generally adopted for the object (and, hence, its skeleton) and for its complement, respectively. In the 3D space, the 26-connectivity and the 6-connectivity are generally used for the object and its complement, respectively.

A second problem, relevant for skeletonization, is related to the nature of the discrete space. In correspondence with regions whose thickness is expressed by an even number of pixels/voxels, the set of centers of maximal balls is 2-pixel/voxel wide. Thus, if a discrete solution to skeletonization is desired, the resulting set (called *nearly-thin skeleton*) can locally be 2-pixel/voxel wide. Alternatively, the nearly-thin skeleton can be reduced to a 1-pixel/voxel thick set by means of a post-processing, generally called *final thinning*, but in this case the complete reversibility is lost, since a number of centers of maximal balls are unavoidably removed from the nearly-thin skeleton. The loss in object recovery starting from the unit-wide S and SS exclusively regards pixels/voxels on the border of the original object and, as such, is generally considered as acceptable. However, if the 3D curve-skeleton CS is computed, reversibility is generally no longer possible, due to the very large number of centers of maximal balls removed from SS to reduce it to CS.

The discrete distance between two elements p and q is the length of a shortest path from p to q. The degree of approximation to the Euclidean distance depends both on the unit moves used to build the path, and on the weights assigned to the unit moves, when these have different Euclidean lengths. A thorough investigation of the distance transforms in 2D and 3D, as well as of weight selection can be found in [5,6]. A good approximation is obtained in 2D by using the weights w_e=3 and w_v=4, for the edge- and the vertex-neighbors respectively, and in 3D by using the weights w_f=3, w_e=4 and w_v=5, for the face-, the edge- and the vertex-neighbors respectively. For the city-block distance in 2D it is w_e=1 and w_v=∞, and for its 3D equivalent distance it is w_f=1, w_e=∞ and w_v=∞.

The *distance transform* DT of the object O with respect to the background B is a replica of the image I, where the elements of O are labeled with their distances from B, computed according to the chosen distance function.

A *center of a maximal ball*, CMB, is an object element whose associated ball is not included by any other single ball in the object. To check this condition, a comparison among the label of the element itself and the labels of its neighbors taking into account the relative weights is sufficient [7]. An object element p is center of a maximal ball if for every neighbor q, it is $q < p+w_i$, where i={f, e, v} depending on whether q is a face-, an edge- or a vertex-neighbor of p. Of course, only edge- and vertex-neighbors exist in 2D.

A *hole* in 2D (a *cavity* in 3D) is a maximal 4-connected (6-connected) component of background elements fully surrounded by object elements.

A *tunnel* exists in a 3D object if a path can be found in the object that cannot be deformed to a single voxel [8].

An object element p is *simple* if the object including p is homotopic to the object deprived of p. Simplicity of p means that the numbers of holes and object components in 2D (the numbers of cavities, object components, and tunnels in 3D) are the same, independently of whether p is in the object or in the background.

An object element is a *border element* in the 2D case (in the 3D case) if it has at least an edge-neighbor (a face-neighbor) in the background. Otherwise, an object element is and *inside element*.

An *end point* is an element of S in 2D (of CS in 3D) having only one neighbor in the skeleton.

3 Skeletonization in 2D

The classical approach to skeletonization is based on the repeated use of topology preserving removal operations. Removal of a pixel p should not create holes, which requires that p has at least an edge-neighbor in the background. Disconnections should not occur, which requires that p has exactly one 8-connected component of object neighbors. Simple arithmetic checks, such as the *crossing number* [9] and the *connectivity number* [10], can be used to respectively count the number of 4-connected components of background elements and the number of 8-connected components of object elements in the neighborhood of p. Finally, in order the skeleton reflects the geometrical properties of the object, removal of p should be permitted only provided that p is not necessary to prevent shortening of peripheral branches in the resulting skeleton.

By following the iterated approach, skeletonization requires a number of iterations proportional to the maximal thickness of the object O. If sequential computers are used, each iteration consists of two sub-iterations, respectively dealing with the identification of the current border, and with the sequential deletion of suitable border pixels, i.e., pixels that, when inspected, are not necessary to preserve topology and are not end points. The process is repeated until no border pixel can be removed from the current border. At this stage of the process, all object pixels are expected to be border pixels; some exceptions are discussed later on in this section.

End point detection is a key task to guarantee isotropic object compression and to avoid unwanted shortening of branches in the resulting skeleton. Every significant protrusion of the object should be mapped into a skeleton branch. To achieve a correct mapping, the tip of each protrusion should be identified and an element of the tip should be prevented from removal, so that it will become an end point in the skeleton. Unfortunately, pixel removal often occurs during a *blind* sequential process, that uses the property satisfied by the end points in the resulting skeleton, i.e., the property of having only one neighbor in the skeleton, as a criterion to detect the end points during the process. Thus, in correspondence with identical configurations end points may be originated or not, depending on the order in which the chosen sequence of removal operations is applied to the object's elements.

To solve this problem, the border configurations corresponding to object protrusions should be identified at the beginning of all iterations of the skeletonization process, before applying the removal operations. Effective criteria can be based on the distance of border elements from the inside of the object. Only border subsets

including elements at distance from the inside larger than a given threshold correspond to significantly elongated object protrusions and, hence, their pixels should be prevented from being removed [11]. Alternatively, effective criteria can be based on the selection and preservation of all centers of maximal balls in DT. In fact, in correspondence with the tip of an object protrusion, a maximal ball certainly exists, whose border fits the border of the object protrusion for a (large) connected part. The center of such a maximal ball can be selected as the end point of the branch into which the protrusion is mapped. In both cases, the components of skeletal pixels found at a given iteration are likely not to be of unit width. The final identification of the end points is actually postponed to the final thinning, when the nearly-thin skeleton is reduced to unit width.

From an operative point of view, the pixels that should not be removed at the k-th step of the iterated skeletonization process (being necessary for topology preservation, or belonging to border subsets corresponding to significant protrusions) are located where the current border C_k is locally nearly-thin, i.e., where C_k folds on itself. These pixels are called *multiple pixels*, since they prevent C_k from being simple.

In the digital plane a simple 8-connected curve is a set of object pixels dividing the background into two connected subsets, respectively called the outside and the inside of the curve. Pixels of a curve are neighbors of both the inside and the outside of the curve. Since both the inside and the outside of the curve consist of background pixels, the 4-connectness is used for both sets. On the other hand, when the curve is actually the border of an object, the inside consists of object pixels while the outside consists of background pixels, so that 8-connectedness and 4-connectedness are respectively used for the inside and the outside.

In [12], we have proved that the border of an object is simple if and only if both conditions below are satisfied for each of its pixels p:

A1 A pair of opposite edge-neighbors of p exists, such that one of these neighbors belongs to the inside of the object and the other neighbor belongs to the background.

A2 A border pixel v, vertex-neighbor of p, does not exist such that the two edge-neighbors of p that are also edge-neighbors of v both belong to the background.

If the border is not simple, then the border pixels for which any of the above conditions does not hold are multiple pixels. These pixels are the only ones that, at each iteration of skeletonization, cannot be removed. An end point detection criterion is not necessary, when using conditions A1-A2. In fact, the pixel(s) delimiting a peripheral (nearly-thin) part of the border protruding from the object are definitely multiple due to condition A1 (they cannot have an edge-neighbor in the inside). Thus, conditions A1-A2 are more powerful within skeletonization than removal operations based on the notion of simple point. In fact, in the latter case an end point detection criterion should be used to avoid unwanted shortening of peripheral branches, since the end points are simple points.

The process terminates when the current border is completely made up of multiple pixels. Ordinarily, this will occur when the current inside is empty. However, in some pathological cases, a border consisting entirely of multiple pixels can exist, while the inside is not empty. Examples of pathological cases are shown in Fig.1. In both cases, the current border consists exclusively of multiple pixels, but the inside is not empty.

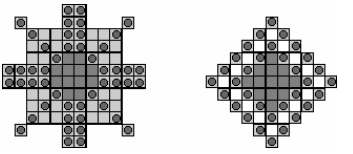

Fig. 1. Object pixels (squares) on a white background. Dots indicate multiple pixels. Light gray and dark gray denote removed object pixels and remaining inside pixels, respectively.

The two conditions A1-A2 can be used more efficiently by employing the city-block distance transform DT of the object during the iterative skeletonization process. In this way, the sub-iteration dealing with current border identification would be no longer necessary, since the DT keeps track of all successive borders. The border of the object at the k-th iteration includes all pixels with distance label k in DT, as well as pixels with distance label smaller than k, if these latter pixels were not removed during previous iterations. Pixels with label greater than k are inside pixels at the k-th iteration. The background is, at all iterations, the set of removed pixels, i.e., the set of 0's. With this interpretation, the skeletonization algorithm can be sketched as follows.

Compute DT. Let m be the maximal distance label in DT
for $k = 1, m$
Remove every pixel p with distance label k, which satisfies conditions A1 and A2, when p is inspected.

It is easy to see that if p is a CMB, condition A1 fails. In fact, no edge-neighbor of p can be labeled more than p, so that p cannot have an edge-neighbor in the inside of the object. Thus, the skeleton includes all the CMBs and complete recovery of the object is possible. The skeleton is nearly-thin, since the set of the CMBs is likely to be 2-pixel thick in some parts.

If the unit-wide S is desired, final thinning is performed by means of topology preserving removal operations. Since inside pixels can exist in the set of skeletal pixels, e.g., for pathological cases or in correspondence of branches crossing each other, only pixels with at least one edge-neighbor in the background are processed, to avoid creation of spurious holes. We suggest a removal operator using the four masks shown in Fig.2. A skeletal pixel p with an edge-neighbor in the background is removed if at least one mask matches the neighborhood configuration of p.

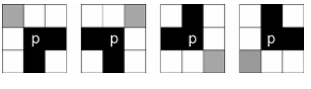

Fig. 2. Masks for final thinning. Black squares and gray squares denote skeletal pixels and background pixels, respectively. White squares are don't care pixels, but among them at least one edge-neighbor of p in each mask must belong to the background.

A post-processing aimed at removal of noisy branches is generally also accomplished (*pruning*). Of course, both final thinning and pruning will remove from the skeleton some CMBs. As a consequence, complete object recovery is not possible,

starting from the unit-wide pruned S. However, the loss in recovery can be kept under control by using context dependent pruning [13].

The performance of the algorithm can be seen with reference to Fig.3. The skeleton has been reduced to unit thickness and pruning of short branches has been applied. The black pixels in proximity of the border of the object in Fig. 3 are those whose recovery is not possible.

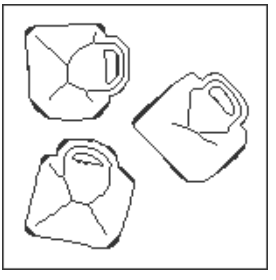

Fig. 3. A mug in three different orientations. The "city-block" skeleton is computed by using conditions A1 and A2.

A different approach to skeletonization can be followed, by exploiting the use of the distance transform. This approach is valid for DTs computed by using any pair of weights w_e and w_v. The DT is interpreted as a landscape, where the label of a pixel indicates its height. Some constraints exist on the heights of neighboring pixels, whose labels can differ only to a limited extent (at most w_e for edge-neighbors, and at most w_v for vertex-neighbors). Thus, the landscape is characterized by gentle slopes converging towards at most two-pixel wide ridges and peaks. The CMBs are located on peaks or ridges and can be detected in one inspection of the DT. Other skeletal pixels detectable in the same inspection of the DT are placed in *saddle configurations*. These pixels have in their neighborhood both (4-connected) components of neighbors with smaller label, and (8-connected) components of pixels with larger label. Indeed, due to the discrete nature of the digital space, in correspondence with parts of the object with even thickness, a special case of saddle configuration occurs, where the pixel p at hand is part of a 2×2 block of pixels all with the same label.

The set consisting of the CMBs and the saddle pixels is not necessarily connected, even for a connected object. Thus, to ensure skeleton connectedness, a path growing process is used. Paths originate from already detected skeletal pixels and proceed in the direction of the distance gradient upwards, until another already detected skeletal pixel is found. When determining the gradient, the weights w_i used to measure the unit moves are taken into account. If p is an already detected skeletal pixel and q is a neighbor of p labeled more than p, the gradient to q from p is $grad\ q = (q\text{-}p)/w_i$. The neighbor of p that maximizes the gradient is the next pixel in the path.

This skeletonization process is computationally very convenient, as it requires only two scans to compute the DT, one scan of the DT plus a path growing process to detect all skeletal pixels, independently of the object's thickness. For more details, see

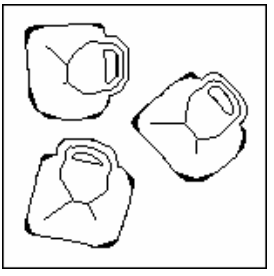

Fig. 4. The skeleton obtained in one inspection of the DT computed with w_e=3 and w_v=4

[13,14]. Also in this case, the skeleton is nearly-thin and the same final thinning already discussed can be employed to obtain the unit-wide skeleton.

An example of the performance of this algorithm by using the DT with weights w_e=3 and w_v=4 is given in Fig.4.

4 Skeletonization in 3D

In 3D, objects can be i) directly reduced to their curve skeletons, e.g., [15,16], or ii) they can be first reduced to their surface skeletons, e.g., [17,18], which are then furthermore compressed to the curve skeletons, e.g., [19,20]. Since object recovery is possible only from the surface skeleton, we prefer algorithms of the latter type.

4.1 Surface Skeletonization

The iterated approach to surface skeletonization requires the identification, iteration after iteration, of the surface bordering the object. Topology preserving removal operations can then be applied. Analogously to the 2D case, where care has to be taken to avoid unwanted shortening of skeleton branches corresponding to significant object protrusions, in 3D removal should not affect peripheral surfaces into which object parts should be mapped. Again, preserving from removal the voxels that are CMBs is a convenient way to achieve this goal.

Also in 3D, topology preserving removal operations can be based on the notion of simple point [21,22], by taking into account that, besides cavities and disconnections, also tunnels have to be considered in the definition of point simplicity. Cavities are not created when removing v from a 26-connected object, if v has at least a face-neighbor in the background. Unfortunately, simple arithmetic checks, analogous to the crossing number or the connectivity numbers available in the 2D case, do not exist to count the number of components in the neighborhood of a voxel v. The number, N_{26}, of 26-connected components of object voxels and the number, N_6, of 6-connected components of background voxels have to be computed by processing the 3×3×3 neighborhood of v. These numbers can be used to determine whether removal of v causes disconnections. Moreover, since removal of v should prevent creation of

tunnels, also the number, N_6^{18}, of 6-connected components of background voxels, having v as face-neighbor and computed in the 3×3×3 neighborhood of v deprived of the eight vertex-neighbors, has to be computed. Due to the heavy computation of N_{26}, N_6 and N_6^{18}, alternative removal operations that limit the use of the above numbers are of interest.

The conditions A1-A2 defined in 2D, respectively involve edge- and vertex-neighbors of a pixel *p*. In 3D, face- and edge-neighbors of a voxel v play the role of the edge- and vertex-neighbors of a pixel p in 2D, respectively. By taking into account that also a third kind of neighbor of a voxel v exists in 3D, the vertex-neighbor, the conditions A1-A2 can be extended to the 3D case, so originating three conditions B1-B3 to detect multiple voxels. A voxel v is not removable if any of the following conditions is satisfied:

B1 No pair of opposite face-neighbors of v exists such that one is an inside object voxel and the other is a background voxel.

B2 There exists a border voxel *e,* edge-neighbor of v, such that the face-neighbors of both e and v are background voxels.

B3 There exists a border voxel *x,* vertex-neighbor of v, such that the six common neighbors of both x and v are background voxels.

In Fig.5, the basic configurations involved in conditions B1-B3 are shown. Of course, rotated and mirrored configurations have to be taken into account.

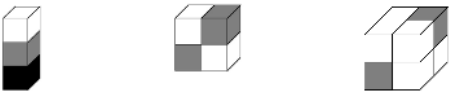

Fig. 5. Voxels involved in Condition B1, left, B2, middle, B3, right. White, gray and black cubes denote background, border and inside voxels, respectively.

If the conditions B1-B3 are sequentially checked for the voxels of the current border, iteration after iteration, a nearly-thin surface skeleton is obtained, as soon as all voxels in the current border are multiple voxels. We point out that the obtained surface skeleton includes the voxels that would be detected as CMBs in the DT computed by using the weights w_f=1, w_e=∝ and w_v=∝. These voxels are in fact detected by condition B1 (analogously to the 2D case, where the CMBs in the city-block DT were detected by condition A1).

If a unit wide surface skeleton is desired, final thinning should be accomplished. Since the only voxels that should be removed by this process are those located where the object is 2-voxel thick in face-direction, we should detect the existence of such subsets in the nearly-thin surface skeleton. To this purpose, we use a 4×1 mask consisting of four voxels aligned along one of the three principal directions (top-bottom or bottom-top, left-right or right-left, and front-back or back-front). In each mask, the two external voxels are background voxels and the two internal ones, say v_1 and v_2, are object voxels. The voxels of the surface skeleton are sequentially inspected

and if the current voxel plays the role of v_1 in the above 4×1 mask, the voxel is removed provided that conditions B2 and B3 do not hold. This is, in fact, sufficient to avoid unwanted reduction of thin peripheral surfaces as well as disconnections.

As an example of the performance of the above algorithm, see Fig.6.

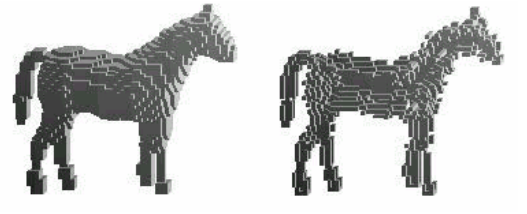

Fig. 6. A 3D object, left, and its surface skeleton, right, obtained by using conditions B1-B3

The distance transform with weights w_f=1, w_e=∝ and w_v=∝ can be used, analogously to the 2D case, to reduce the computational effort of the above algorithm. In fact, by computing the DT all the borders that will successively characterize the object are available. As in 2D, at the *k*-th iteration, the border consists of voxels with distance label *k,* as well as of voxels with smaller distance label that have been prevented from being removed at previous iterations. The inside includes all voxels with distance label larger than *k,* and the background the voxels set to 0.

Efficient methods to furthermore exploit the use of the DT with the purpose of computing the surface skeletons in a fixed and small number of scans have been suggested only for the DT computed with weights w_f=1, w_e=∝ and w_v=∝ [23].

4.2 Curve Skeletonization

Although curve skeletonization is not reversible since a number of CMBs have unavoidably to be removed from the surface skeleton *SS* to obtain the curve skeleton *CS*, this process is still of interest. In fact, the curve skeleton is a useful representation of the object, provided that it at least maintains some shape information.

To obtain the curve skeleton, an iterated removal process can be applied to the surface skeleton to remove voxels located along the border of the surface skeleton. Thus, a classification of the voxels of the surface skeleton is necessary [20]. Four types of voxels can be identified: internal, edge, curve, and junction voxels. Obviously, internal voxels should not be removed, to avoid creation of spurious tunnels. Curve voxels have to be preserved from removal, since they already constitute parts of the desired curve skeleton. Junction voxels have also to be preserved to avoid disconnections. Thus, the only voxels that should undergo the removal process are the edge voxels. Due to voxel removal during an iteration, some voxels initially classified as internal or junction voxels are likely to become edge voxels at the successive iteration and, as such, should undergo the removal process. However, the basic idea behind curve skeletonization is to postpone as much as possible removal of the voxels classified as junction voxels in the initial surface

skeleton, since junctions between different subsets of the surface skeleton retain significant shape information. Thus, curve skeletonization is performed in two phases. During the first phase, removal of voxels classified as junction voxels in the initial surface skeleton is prevented, even if those junction voxels are transformed into edge voxels at some iteration. During the second phase, all current edge voxels are candidate to removal.

Topology preserving removal operations, based on the computation of N_{26}, N_6 and N_6^{18}, are used during both phases. The obtained curve skeleton is at most two-voxel thick, is centered within the surface skeleton, and is homotopic to the surface skeleton (and, hence, is homotopic to the original object). No end point detection criterion is necessary. In fact, end points are automatically preserved, due to their classification as curve voxels whose removal is always prevented.

To reduce the nearly-thin curve skeleton to unit width, final thinning has to be applied. The 4×1 mask introduced in Section 4.1 can be used to identify voxels candidate to removal. These voxels are removed only if this does not alter topology. Actually, two-voxel thickness of the curve skeleton can also be due to pairs of voxels, which are edge-neighbors of each other. Thus, final thinning makes use of different strategies, depending on whether two-voxel thickness is due to pairs of skeletal voxels that are face- or edge-neighbors of each other. Finally, pruning some of the peripheral branches is generally done [24]. Pruning is performed by tracing any peripheral branch and by removing its voxels, provided that the branch includes a small percentage of *significant* voxels, where the notion of significance is still based on the voxel classification accomplished on the surface skeleton.

In Fig.7, a 3D object, its surface skeleton, the nearly-thin curve skeleton, and the unit-wide curve skeleton before and after pruning are shown.

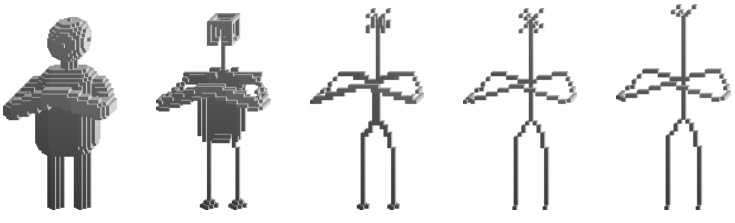

Fig. 7. From left to right: a 3D object, the surface skeleton, the nearly-thin curve skeleton, the unit-wide curve skeleton, and the pruned unit-wide curve skeleton

5 Concluding Remarks

Skeletonization is a convenient process to obtain an object representation with reduced cardinality and suited for image analysis. In the 2D case, the skeleton is union of arcs and curves placed symmetrically within the object. It has been used in many applications, e.g., in optical character recognition. In the 3D case, the skeleton can be either a surface skeleton, or a curve skeleton. The latter can be computed only for objects without cavities. The 3D skeleton is union either of surfaces and curves, or

of curves. It has been used in a number of applications, e.g., for the analysis of angiographies starting from magnetic resonance images.

In this paper, approaches to 2D and 3D skeletonization have been presented and algorithms to efficiently compute the skeleton have been briefly illustrated. In particular, it has been shown that the computational load can be significantly reduced when using the distance transform of the object.

References

1. H. Blum, "A transformation for extracting new descriptors of shape", in W. Wathen-Dunn, ed., *Models for the Perception of Speech and Visual Form*, M.I.T. Press, Cambridge, MA, 1967, 362-380.
2. H. Blum, "Biological shape and visual science", *J. Theor. Biol.*, 38, 205-287, 1973.
3. T.Y. Kong and A. Rosenfeld eds., *Topological Algorithms for Digital Image Processing,* Elsevier 1996.
4. C.H. Chen, P.S.P. Wang, Eds., *Handbook of Pattern Recognition and Computer Vision*, World Scientific, Singapore, 2005, chapter 2.3.
5. G. Borgefors, "Distance transformation in digital images", *Comput. Vision Graphics Image Process.*, 34, 344-371, 1986.
6. G. Borgefors, "On digital distance transform in three dimensions", *Computer Vision and Image Understanding*, 64-3, 368-376, 1996.
7. C. Arcelli, G. Sanniti di Baja, "Finding local maxima in a pseudo Euclidean distance transform", *Comput. Vision Graphics Image Process.*, 43, 361-367, 1988.
8. T.Y.Kong, "A digital fundamental group", *Computers and Graphics*, 13, 159-166, 1989.
9. C.J.Hilditch, "Linear skeletons from square cupboards", in B.Meltzer and D.Michie eds., *Machine Intelligence IV*, Edinburgh University Press, UK, 1969, 403-420.
10. S.Yokoi, J.I.Toriwaki, T.Fukumura, "An analysis of topological properties of digitized binary pictures using local features", *Comput. Graphics Image Process.* 4, 63-73, 1975.
11. C.Arcelli, G. Sanniti di Baja, "A thinning algorithm based on prominence detection", *Pattern Recognition*, 13-3, 225 235, 1981.
12. C. Arcelli, G. Sanniti di Baja, "A contour characterization for multiply connected figures", *Pattern Recognition Letters,* 6, 245-249, 1987.
13. G. Sanniti di Baja, E. Thiel, "Skeletonization algorithm running on path-based distance maps", *Image and Vision Computing,*14, 47-57, 1996.
14. C. Arcelli, G. Sanniti di Baja, "Euclidean skeleton via centre-of-maximal-disc extraction", *Image and Vision Computing,* 11, 163-173, 1993.
15. K. Palagyi, A. Kuba, "A parallel 3D 12-subiteration thinning algorithm", *Graphical Models and Image Processing*, 61, 199-221, 1999.
16. P.K. Saha, B.B. Chaudhuri, D.D. Majumder, "A new shape preserving parallel thinning algorithm for 3D digital images", *Pattern Recognition*, 30-12, 1939–1955, 1997.
17. G. Borgefors, I. Nyström, G. Sanniti di Baja, "Surface skeletonization of volume objects" in P. Perner et al. eds., *Advances in Structural and Syntactical Pattern Recognition*, LNCS 1121, Springer-Verlag, 251-259, 1996.
18. G. Bertrand., "A parallel thinning algorithm for medial surfaces", *Pattern Recognition Letters,* 16, 979–986, 1995.
19. G. Borgefors, I. Nyström, G. Sanniti di Baja, "Computing skeletons in three dimensions", *Pattern Recognition*, 32-7, 1225-1236, 1999.
20. S. Svensson , I. Nyström, G. Sanniti di Baja, "Curve skeletonization of surface-like objects in 3D images guided by voxel classification", *Pattern Recognition Letters*, 23-12, 1419-1426, 2002.

21. P.K.Saha, B.B.Chaudhuri, “Detection of 3D simple points for topology preserving transformations with application to thinning”, *IEEE Trans. PAMI*, 16, 1028-1032, 1994.
22. G.Bertrand, G.Malandain, “A new characterization of three-dimensional simple points”, *Pattern Recognition Letters*, 15, 169-175, 1994.
23. G.Sanniti di Baja, S.Svensson, “Surface skeletons detected on the D^6 distance transform”, in F.J. Ferri et al. eds., *Advances in Pattern Recognition*, LNCS 1121, Springer-Verlag, 387-396, 2000.
24. S. Svensson, G. Sanniti di Baja, “Simplifying curve skeletons in volume images”, *Computer Vision and Image Understanding*, 90, 242-257, 2003.