Skeleton Pruning Based on Elongation and Size of Object's Limbs and Boundary's Convexities

Luca Serino(✉) and Gabriella Sanniti di Baja

Institute for High Performance Computing and Networking, CNR, Naples, Italy
{luca.serino,gabriella.sannitidibaja}@cnr.it

Abstract. We present a new pruning method able to remove peripheral branches of the skeleton of a 2D object without altering more significant branches. Pruning criteria take into account elongation and size of the object's parts associated with skeleton branches. Only peripheral branches associated with scarcely significant object's limbs and boundary's convexities are removed, so that the object can be recovered satisfactorily starting from the pruned skeleton. Since by removing peripheral branches, new peripheral branches can be created, pruning is iterated until the skeleton structure becomes stable. The algorithm does not require fine tuning of the parameters and the obtained results are satisfactory.

Keywords: Shape representation · Shape analysis · Skeleton · Pruning

1 Introduction

The skeleton is a well known representation system useful in the framework of shape analysis. A wide literature is available as concerns different skeletonization methods, devised in the continuous and in the digital space, and the use of the skeleton in several application fields [1]. Ideally, the skeleton is characterized by unit thickness, centrality, homotopy, recoverability and significance. This means that the skeleton should consist exclusively of curves placed in the middle of the object, and should be characterized by the same topological features as the object; moreover, skeleton pixels should be labeled with their distance from the complement of the object so that the object can be recovered by the union of the discs centered on the skeleton pixels and having radii equal to the corresponding distance values; finally skeleton branches should be found only in correspondence of significant limbs and strong boundary convexities of the object. Actually, whichever skeletonization algorithm is considered, the obtained skeleton S unavoidably includes a number of peripheral branches not all obtained in correspondence of individually meaningful object protrusions or boundary's convexities, thus making skeleton pruning an indispensable step for any skeletonization algorithm.

Pruning can be accomplished by preventing the creation of non significant branches via a preliminary filtering process that identifies suitable anchor points on the tips of significant object's convexities from which skeleton branches will originate [2, 3]. More typically, pruning is accomplished during a post-processing phase aimed at removing scarcely significant branches.

M. Mendoza and S. Velastín (Eds.): CIARP 2017, LNCS 10657, pp. 609–617, 2018.
https://doi.org/10.1007/978-3-319-75193-1_73

In the literature, different criteria have been suggested to distinguish branches corresponding to significant parts of the object from those whose removal does not affect the representative power of the skeleton. See for instance [4], where pruning criteria based on propagation velocity, maximal thickness, radius function, axis arc length, and ratio between boundary and axis length are presented. A pruning criterion based on the number of slices of object pixels that would not be recovered starting from the pruned skeleton is discussed in [5]. Other pruning methods involve contour partitioning via discrete curve evolution and bending potential ratio [6, 7], where pruning is either performed during a post-processing step, or is embedded into the skeleton computation process. Sequential and parallel modalities to perform pruning are discussed in [8], while criteria to remove concatenations of branches without altering the topology of S or reducing its representative power are described in [9].

In this paper, we present a pruning method based on elongation and size of object's protrusions and boundary convexities. The method can be applied to any skeleton, provided that its pixels are labeled with their distance from the complement of the object. In this paper we refer to the skeleton computed by the algorithm [5], where the <3, 4> distance [10] is used to label skeleton pixels. Elongation and size of the object part associated with a skeleton branch are evaluated by resorting to polygonal approximation of the skeleton branch in a 3D space, where the three coordinates of any skeleton pixel are its spatial coordinates and distance label. The criterion adopted to evaluate the goodness of the suggested method is based on the reconstruction ability of the skeleton. The higher is the percentage of pixels of the input object recovered by the pruned skeleton, the better the object is represented by the pruned skeleton.

2 Basic Notions

We work with binary images, where the object consists of the pixels with value 1, while the background consists of the pixels with value 0. The 8-connectedness and the 4-connectedness are used for the object and the background, respectively. Since the skeleton S is a subset of the object, the 8-connectedness is used also for S.

Given two pixels p and q, their <3, 4> distance $d(p, q)$ is the length of a minimal 8-connected path linking p to q, where unit moves towards horizontal/vertical neighbors and diagonal neighbors along the path are respectively weighted 3 and 4.

For the sake of simplicity, let us suppose that only one 8-connected object exists in the image at hand, so that its skeleton S consists of exactly one 8-connected component. Pixels of S are classified as end points, normal points, or branch points, depending on whether they have one, two, or more than two neighbors in S.

Each branch of S, which is a curve entirely consisting of normal points with the exception of the two pixels delimiting the curve, which can be end points or branch points, maps a region of the input object. A branch is termed peripheral branch if one delimiting pixel is an end point and the other is a branch point. To avoid altering the topology of S, only peripheral branches can be pruned. Thus, skeletons rid of peripheral branches do not undergo pruning.

When a skeleton branch is pruned, its delimiting branch point is not removed; it can be transformed into a new end point or into a normal point, or can maintain the status of branch point, depending on whether all or only part of the peripheral branches sharing it are pruned. Once the initial peripheral branches of S have been checked against the pruning criteria and possibly removed, new peripheral branches might have been created in S due to the transformation of some branch points into end points or normal points. Then, pruning is applied again. The process is iterated until the skeleton becomes stable.

3 The Method

At each iteration, the following main tasks are performed: (1) polygonal approximation of peripheral skeleton branches, (2) computation of elongation and size of the corresponding object's parts, and (3) removal of scarcely significant branches. Pruning is iterated until the skeleton reaches a stable structure.

Polygonal approximation

We resort to the split type approach [11] to compute the polygonal approximation of peripheral skeleton branches. The extremes of the current skeleton branch are taken as the two starting vertices from which the process recursively splits the branch. The Euclidean distance of all normal points of the skeleton branch from the straight line joining the two vertices is computed. The point at the largest distance is taken as a new vertex, provided that such a distance overcomes an a priori fixed threshold θ. Otherwise the process terminates. When a new vertex is detected, the branch is divided into two curves, to each of which the split type algorithm is applied. Splitting continues as far as new vertices are detected. At the end of the process, the skeleton branch is approximated by a number of curves that, in the limits of the adopted tolerance, are straight segments and is represented by the ordered sequence of the detected vertices. To explain the reason for which polygonal approximation is performed in a 3D space, where the three coordinates of any skeleton pixel are its spatial coordinates and distance label, let us consider the object in Fig. 1 left and its skeleton S in Fig. 1 middle, where different colors denote different distance values. We may note that the geometry of S reflects the geometry of the object: curvature changes along the skeleton correspond to curvature changes along the object boundary of the object. Thus, polygonal approximation of S in the 2D space would divide it into straight segments corresponding to parts of the object whose boundary is characterized by the absence of curvature changes. We may also note that the different distance values of the pixels of S take into account the changes in width of the object. Thus, by approximating S in the 3D space, we divide it into straight segments along which distance labels are either constant or monotonically increase/decrease. Each so found segment is interpreted as the spine of a simple region whose boundary is rid of curvature changes and whose width is either constant or linearly increases/decreases. From an operative point of view, the computation of the Euclidean distance d of a point C from the straight line joining two points A and B in the 3D space is done by using the following expression:

$$d^2 = ||\mathrm{AC}||^2 - \mathrm{P_{ABC}} * \mathrm{P_{ABC}}/||\mathrm{AB}||^2 \quad (1)$$

where $||\mathrm{AB}||$ is the norm of the vector AB, and $\mathrm{P_{ABC}}$ is the scalar product between vectors AB and AC. We have experimentally found that to obtain a quite faithful approximation of S with a reasonably small number of vertices, the best value for the threshold is in the average $\theta = 3.5$. Such a value has been used for all the examples shown in this paper. As an example, see Fig. 1 right, where the found vertices are shown in black.

Fig. 1. From left to right, an object, its skeleton (different colors denote different distance values), and the vertices (black) resulting in the skeleton after polygonal approximation in 3D. (Color figure online)

Size and elongation

The advantage provided by polygonal approximation is that elongation and size of the object's parts associated with skeleton branches can be computed immediately once spatial coordinates and distance labels of the vertices are available. In fact, a simple region is shaped either as a rectangle (Fig. 2 middle left), when the distance labels are constant along the spine (Fig. 2 left), or as a trapezium (Fig. 2 right), when distance labels linearly increase/decrease along the spine (Fig. 2 middle right), and is delimited by two half discs centered on the two vertices of the spine. A concatenation of simple regions is associated to a skeleton branch approximated by a number of segments.

Fig. 2. Straight line skeleton segments and their corresponding simple regions.

We remark that any vertex along a skeleton branch (except the end point and the branch point) is common to two successive spines. We also remark that pruning does not remove the branch point delimiting a skeleton branch. Thus, the evaluation of the size of the object's part that would be lost by removing the corresponding skeleton branch is done as follows. For each simple region in the concatenation forming the object's part associated with the *i-th* peripheral skeleton branch, the half disc centered

on the more external vertex is added to the area of the rectangle/trapezium, while the half disc centered on the more internal vertex is subtracted. The total area A_i associated to the *i-th* skeleton branch is the sum of the so computed contributions in correspondence with the spines polygonally approximating the *i-th* skeleton branch.

The elongation E_i of the object's part that would be lost by removing the *i-th* skeleton branch is computed by adding to the radius of the disc centered on the end point of the *i-th* skeleton branch the height of each rectangle/trapezium in the concatenation of simple regions and by subtracting the radius of the disc centered on the branch point. The height of any rectangle/trapezium is computed as the distance between the two vertices delimiting the corresponding spine.

Pruning

To decide whether the *i-th* peripheral skeleton branch can be pruned, we should compare the elongation E_i and the size A_i of the corresponding object's part with two thresholds. We automatically compute the value for the threshold on elongation in terms of the average elongation AvE of all the skeleton branches that result to be peripheral at the current iteration of the process, and regard the *i-th* peripheral branch prunable with respect to elongation if the following condition is satisfied:

$$E_i < e \times AvE_i \tag{2}$$

where e is a constant.

We set the value of the threshold on size in terms of the ratio between the size A_i of the object part that would be lost by removing the *i-th* skeleton branch and the size D_i of the disc centered on the branch point of the *i-th* skeleton branch, and regard the *i-th* skeleton branch prunable with respect to size if the following condition is satisfied:

$$A_i/D_i < a \tag{3}$$

where a is a constant.

We take into account that often S is affected by a large number of "short" noisy peripheral branches, whose contribution to AvE significantly reduces the real significance of this parameter. See for example Fig. 3 left. Thus, at each iteration of pruning we perform a pre-processing step to remove "short" noisy branches. We regard a skeleton branch as "short" if it consists of at most four pixels. A close up of a portion of S is given in Fig. 3 middle, where "short" branches are shown in red. This choice is done since any such a branch would be definitely approximated by a single spine, so that during pre-processing we don't perform polygonal approximation of "short" branches and directly take their delimiting end points and branch points as the extremes of the corresponding spines. Obviously, a "short" branch is not necessarily noisy and might still be important for a quite faithful object recovery. Thus, we consider the *i-th* "short" branch as noisy and remove it only if condition (3) is satisfied for it. We point out that a number of pixels different from four can characterize the length of "short" noisy branches, depending e.g., on acquisition noise and image resolution. However, even if some "short" branches are longer than four pixels and hence are not removed during pre-processing, we still considerably trim S so that the successive computation of AvE is done in a more reliable manner.

After "short" noisy branches have been pruned, polygonal approximation of the current peripheral branches of S is done and E_i, A_i and D_i are computed for each branch. The average elongation AvE of all peripheral branches is also computed.

In general, S consists both of noisy peripheral branches and of significant peripheral branches. If the majority of peripheral branches are noisy, a larger value for the threshold on elongation can be used, while a smaller value is necessary if S has a limited number of noisy branches, so as to avoid pruning together with noisy branches also some significant branches. We evaluate whether a smaller or a larger threshold value should be used by exploiting the criterion adopted to measure the goodness of pruning, which is based on the reconstruction ability of the skeleton. If at the beginning of a given iteration of pruning the number of pixels recovered by the un-pruned skeleton, *#Unpr*, and the number of pixels recovered by S if all its peripheral branches are removed, *#Prun*, have similar size, we could argue that all peripheral branches of S are noisy. Clearly, in general not all peripheral branches of S are noisy, so that we need to set a threshold τ_{iter} on the ratio *#Prun*/*#Unpr* to decide whether we can use larger threshold value. To take into account that pruning is iterated until prunable branches are found, we also compare the ratio between *#Prun* and the number of pixels in the input object, *#Obj*, with another, smaller, threshold τ_{obj}. Then, if *#Prun*/*#Obj* > τ_{obj} and *#Prun*/*#Unpr* > τ_{iter}, a larger value for the threshold on elongation can be used by setting $e = e_1 > 1$ in condition (2). Otherwise, $e = 1$.

We have experimentally found that in the average satisfactory results are obtained by setting $\tau_{obj} = 0.80$, $\tau_{iter} = 0.85$, $e_1 = 6$ and $a = 0.6$.

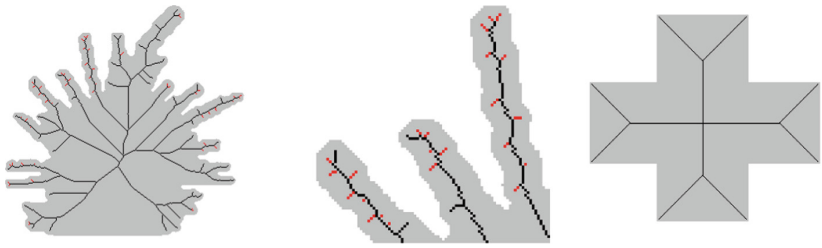

Fig. 3. Left, a noisy skeleton with many "short" peripheral branches; middle, a close up with "short" branches in red; right, a skeleton with peripheral branches with similar elongation. (Color figure online)

Finally, if the peripheral branches of S are all characterized by very similar elongation (see for example the skeleton in Fig. 3 right), we regard all the skeleton branches as significant and no pruning at all is accomplished. We have experimentally verified that if all peripheral branches are characterized by elongation that differs from AvE for at most 10%, pruning should not be accomplished.

Summarizing, at each iteration the following tasks are performed:

(1) Peripheral branches including at most four pixels are identified and each of them is removed if $A_i/D_i < a$;
(2) Peripheral branches are identified and undergo polygonal approximation, during which for each of them elongation E_i, size A_i and area of the disc D_i for the associated object's part are computed. The average elongation AvE is also computed;
(3) If the elongation of all peripheral branches differs from AvE for at most 10%, all peripheral branches are significant and pruning terminates;
(4) If $\#Prun/\#Obj > \tau_{obj}$ and $\#Prun/\#Unpr > \tau_{iter}$, the *i-th* peripheral branch is pruned provided that $A_i/D_i < a$ and $E_i < e_1 \times AvE_i$, else the *i-th* peripheral branch is pruned provided that $A_i/D_i < a$ and $E_i < AvE_i$.

The tasks (1)–(4) are repeated until peripheral branches are pruned. In general, at most four iterations are enough to obtain the final result. An example where skeleton stability is reached in two iterations is shown in Fig. 4.

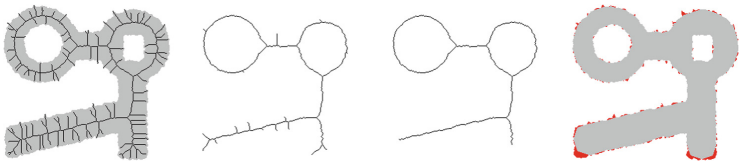

Fig. 4. From left to right, the un-pruned skeleton superimposed on the input object, the skeleton resulting after the 1st and after the 2nd iteration, and the object recovered by the pruned skeleton (object pixels that are not recovered by the pruned skeleton are shown in red). (Color figure online)

We have tested our pruning method on a large set of binary images, taken from shape repositories such as tosca.cs.technion.ac.il/book/resources_data.html, cs.rug.nl/svcg/Shapes/SkelBenchmark, vision.lems.brown.edu/faculty/kimia, cs.toronto.edu/~dmac/ShapeMatcher/. We point out that our method is robust in case of noisy objects and when the input object appears at a different scale or in a different pose. In fact, the average value of the ratio between input pixels recovered by the pruned skeleton and input pixels recovered by the un-pruned skeleton is about 0.96, showing that the representative power of the pruned skeleton is not significantly modified with respect to that of the un-pruned skeleton.

A few examples showing the performance of our pruning method can be appreciated with reference to Fig. 5. We remark that the same values of the parameters involved in the pruning algorithm have been computed for all the images independently of image resolution, object size and position, and noise.

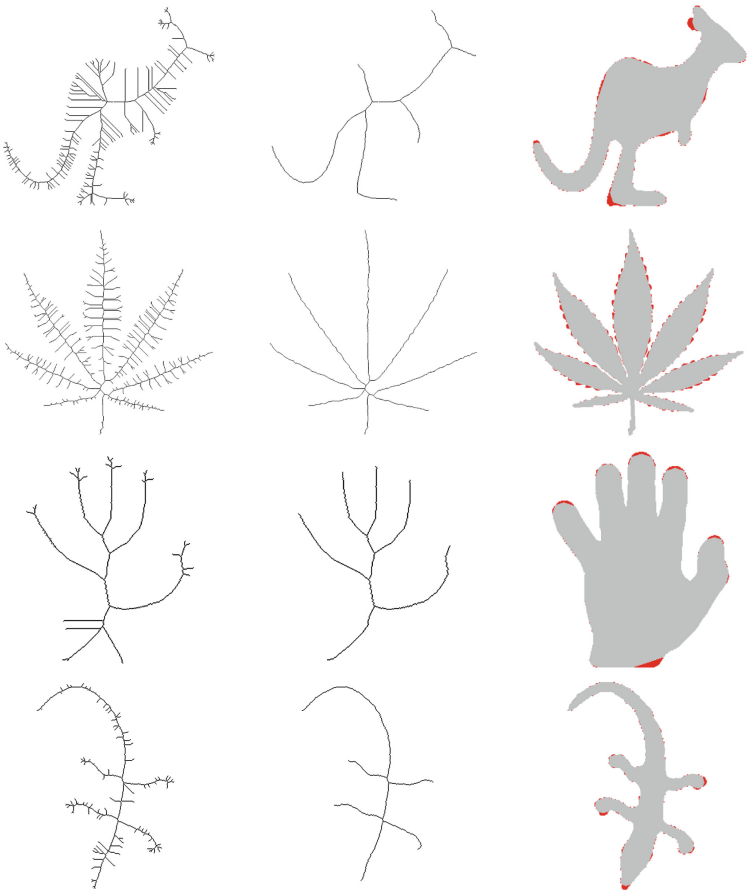

Fig. 5. From left to right, un-pruned skeletons, pruned results, reconstructions.

4 Conclusion

We have introduced a pruning method that removes scarcely significant branches of the skeleton of 2D objects without altering significant branches. The two main pruning criteria take into account elongation and size of the object's parts associated with skeleton branches. To guarantee topology preservation, only peripheral branches are possibly removed. Since removal is done only for branches associated with scarcely significant object's parts, the object can be recovered satisfactorily starting from the pruned skeleton. Pruning is iterated since removal of peripheral branches may create new peripheral branches. The algorithm does not require fine tuning of the parameters and the obtained results are satisfactory.

References

1. Siddiqi, K., Pizer, S.M. (eds.): Medial Representations: Mathematics, Algorithms and Applications. Springer, Berlin (2008). https://doi.org/10.1007/978-1-4020-8658-8
2. Serino, L., Sanniti di Baja, G.: Selecting anchor points for 2D skeletonization. In: Kamel, M., Campilho, A. (eds.) ICIAR 2011. LNCS, vol. 6753, pp. 344–353. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21593-3_35
3. Postolski, M., Couprie, M., Janaszewski, M.: Scale filtered Euclidean medial axis and its hierarchy. Comput. Vis. Image Underst. **129**, 89–102 (2014)
4. Shaked, D., Bruckstein, A.M.: Pruning medial axes. Comput. Vis. Image Underst. **69**(2), 156–169 (1998)
5. Sanniti di Baja, G.: Well-shaped, stable and reversible skeletons from the (3, 4)-distance transform. VCIR **5**, 107–115 (1994)
6. Bai, X., Latecki, L.J., Liu, W.-Y.: Skeleton pruning by contour partitioning with discrete curve evolution. IEEE Trans. PAMI **29**(3), 449–462 (2007)
7. Shen, W., Bai, X., Hu, R., Wang, H., Latecki, L.J.: Skeleton growing and pruning with bending potential ratio. Pattern Recogn. **44**, 196–209 (2011)
8. Frucci, M., Sanniti di Baja, G., Arcelli, C., Cordella, L.P.: On the strategy to follow for skeleton pruning. In: De Marsico, M., Fred, A. (eds.) ICPRAM 2013, pp. 263–266. SCITEPRESS, Lisboa (2013)
9. Serino, L., Sanniti di Baja, G.: A new strategy for skeleton pruning. Pattern Recogn. Lett. **76**, 41–48 (2016)
10. Borgefors, G.: Distance transformations in digital images. Comput. Vis. Graph. Image Process. **34**(3), 344–371 (1986)
11. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. CGIP **1**, 244–256 (1972)