Computers & Graphics 36 (2012) 477–487

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SMI 2012: Full Paper

Skeleton pruning by contour approximation and the integer medial axis transform☆

Andrés Solís Montero*, Jochen Lang[1]

School of Electrical Engineering and Computer Science (EECS), University of Ottawa – Universite d'Ottawa, 800 King Edward Avenue Ottawa, On., Canada K1N 6N5

ARTICLE INFO

Article history:
Received 2 December 2011
Received in revised form
14 March 2012
Accepted 17 March 2012
Available online 28 March 2012

Keywords:
Skeleton
Medial axis
Feature transform
Thinning
Pruning
Contour approximation

ABSTRACT

We present a new shape skeleton pruning algorithm based on contour approximation and the integer medial axis. The algorithm effectively removes unwanted branches, conserves the connectivity of the skeleton and respects the topological properties of the shape. The algorithm is robust to significant boundary noise and to rigid shape transformations, it is fast and easy to implement. High accuracy reconstruction of the shape is possible from the generated skeleton by means of the integer medial axis transform. Our algorithm also produces a vector representation of the skeleton. We compare our algorithm with state-of-the-art techniques for computing stable skeleton representations of shapes including pruning. We test and compare our solution using the MPEG-7 CE Shape-1 Part B dataset looking for skeleton connectivity, complexity, parameter selection, and accuracy/quality of the outcome. The experimental results show that our solution outperforms existing solutions according to these criteria.

1. Introduction

Since Blum's [1] introduction of the medial axis, medial representations of shapes have been studied extensively in the literature [2]. Several authors have covered definitions, mathematical properties and their relation to the object boundaries, therefore their representation in 2D and 3D [2,3]. The medial axis can be seen as the set of points with more than one closest equidistant point to the shapes boundary. The medial axis of a shape gives a simple and compact representation of the shape. The medial axis coupled with the distance from the boundary, make it possible to reconstruct the original shape. Geometrically complex shape boundaries have a complex medial axis, while simplified shape boundaries can also be described with a simpler medial axis. The medial axis of the simplified shape can be considered a simplified skeleton representation of the complex shape. Often a simplified version of the shape is useful, e.g., in several applications in computer vision, image analysis, and digital image processing. Some of these applications include shape matching [4,5], shape segmentation [6], optical character recognition [7,8], and fingerprint recognition [9], among others.

One approach to shape simplification is to prune branches from a complex skeleton. Skeleton branches can be pruned by many different means, e.g., by distance or angle thresholds [10], by shape approximations [11], by smoothing of boundaries, or by growing and shrinking the shape's boundary [12–14]. The definition of a desirable pruning outcome is not universal and hence an automatic pruning solution does not exist to the best of our knowledge. Some of the existing algorithms create critical geometrical and topological changes to the skeleton's curves, creating more noise points, shrinking the final skeleton, disconnecting a skeleton's curves or removing some important branches of the skeleton. All these drawbacks decrease the effectiveness of the analysis of the shape in several applications [14]. Unfortunately, skeletonization algorithms of discrete shapes are particularly sensitive to noise and topological changes [11] because the quantized pixel locations cause excessive branching (see Fig. 1). In this paper we center our attention on finding a skeleton for shapes represented by two-dimensional binary raster images, i.e., the input to our algorithm is a discrete shape representations with its inherently limited spatial resolution.

The desired outcome of skeleton pruning is often application dependent, although all pruning techniques of shape skeletons aim at the simplification of the medial axis. Some output skeletons that are shifted far from the medial axis to simplify their representation [12,13,15,16], others are discrete approximations of the medial axis [10,17,18,16], connectivity [11,19], topology [18], complexity [10] and automation [15,20,21] are

☆If applicable, supplementary material from the author(s) will be available online after the conference. Please see http://dx.doi.org/10.1016/j.cag.2012.03.029.

* Corresponding author. Tel.: +1 613 562 5800x6241.
E-mail addresses: asolis@eecs.uottawa.ca (A. Solís Montero), jlang@eecs.uottawa.ca (J. Lang).

[1] Tel: +1 613 562 5800x6317; fax: +1 613 562 5175.

0097-8493/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2012.03.029

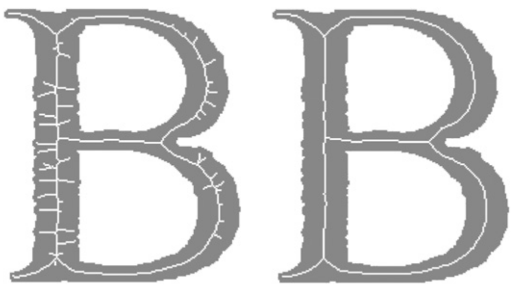
Fig. 1. Skeleton of a noisy image and pruned solution.

other goals. The objective of this paper is a skeletonization and pruning solution that preserves the main features of the shape and its topological properties. Our contributions are a novel pruning technique that conserves connectivity of the pruned skeleton and outputs a subset of the integer medial axis of the input shape and is suitable for automatic real-time applications. In our experimental validation, our pruning algorithm outperforms existing solutions in terms of stability, i.e., the result is more stable than existing solutions even if the shape undergoes a rigid image transformations and if different input parameters are selected. Our method uses only two intuitive threshold parameters: boundary approximation error and a scale parameter for the final pruning. Because our algorithm computes a skeleton that is a discrete approximation of the medial axis, it allows reconstruction of the simplified shape from the skeleton.

Our skeletonization and pruning solution described in Section 3 is a combination of four steps. First, the contours of the shape are approximated by simple piecewise curves; we discuss lines and cubic shape approximation in Section 3.1. Second, we compute the skeleton with the integer medial axis (IMA) transform and prune with the help of the contour approximation (see Section 3.2). Third, we introduce our novel criterion to prune noisy branches in Section 3.3. Finally, the skeleton points are vectorized using piecewise cubic curves as described in Section 3.4. The algorithm achieves our above stated goals on the MPEG-7 CE Shape-1 Part B dataset of 1400 images and performs better than competing approaches with respect to the objective criteria detailed in Section 4. The results obtained with our algorithm are stable with respect to boundary noise and rigid transformations (i.e., noisy and rotated versions of an image produce the same clean skeleton representation as the original image). Noisy branches are removed without loss of skeleton connectivity and the pruned skeleton conserves the topological properties of the shape. The algorithm's complexity is $O(kn + N)$; where k is the number of knots detected during the contour approximation, n is the contour length and N is the image size (see Section 3.5). It is therefore an efficient (and practically fast) solution. The parameter choice has little influence on outcome, allowing the use of the same parameters for the complete test dataset of 1400 images. (We used a distance threshold $T=25$ for the boundary approximation error and a scale factor $S=1.2$ in the final pruning).

2. Literature review

Many different algorithms exist to compute and prune a skeleton of a binary shape. The main categories of different skeletonization algorithms are: shape thinning [15,20–22,16], Voronoi-based analysis [11,14,23–25] and algorithms based on the feature transform [10,11] or on the distance transform [17,18,22].

In shape thinning [15,20,21], pixels are peeled from the boundary shape according to a set of rules relating to a pixel's neighborhood. The process is repeated until there are no more pixels to remove from the shape and the shape skeleton remains. These solutions conserve the connectivity of the final skeleton representation and the topological properties of the shape. They also work automatically because there is no input parameter selection. On the other hand, although their output is a skeleton with many desirable properties, they fail to compute the actual medial axis. As a result, most of thinning algorithms do not save the distance to the boundary [26] and shape reconstruction from the skeleton is not possible [11,15]. Also, these algorithms are often slow because of the re-iterative process of contour detection and pixel removal.

Another type of algorithm is based on Voronoi diagrams [27,28,16]. These solutions sample points on the boundary and compute the Voronoi regions of these points. The boundary of the regions inside the binary shape are an approximation of the skeleton of the shape. The skeleton approximation will become closer to the exact solution as the number of boundary points goes to infinity. Voronoi-based algorithms are typically quite complex because selecting the correct samples of the boundary is involved. As a result many algorithms in this category are slow [11,27]. Martinez et al. [23] propose two variants to compute the Voronoi regions that simplify the boundary sampling problem but they still need a post-processing step to prune their skeleton. Liu et al. [16] propose the use of a Voronoi diagram to create their extended grassfire transform keeping track of the distance of each Voronoi vertex to the boundary. Besides desirable theoretical properties of their solution, it is an interesting mixture between thinning, Voronoi-based, and feature transform based algorithms but initial sample selection influences the outcome.

The third type is based on the mapping of the feature transform [10,29] or distance transform [17,18,22]. The feature transform computes the closest boundary point for each pixel inside a digital shape and the distance transform computes the distance to the nearest boundary pixel. These algorithms can be fast, they detect the medial axis of the digital shape accurately, and shape reconstruction can be easily achieved. On the other hand, noise sensitivity, and pruning techniques that conserve connectivity and skeleton topology are challenging in these types of algorithms. Sanniti di Baja and Thiel [18] present a topology preserving pruning solution to this problem. They prune peripheral skeleton branches based on a comparison between the shape reconstructed with the inverse distance transform applied to the skeleton before and after removal of a branch. They only remove complete peripheral branches if the change in shape is below a threshold related to the length of the branch.

Because of excessive branching of skeletons for noisy digital shapes, pruning is an essential part of most practical skeletonization algorithms. Many different pruning techniques have been proposed but they fall mainly in two categories: those which preprocess the shape's boundary [12,13,26,30,31] (e.g., boundary smoothing) and those that aim to remove unwanted skeleton points depending on some classification criteria [10,11,26,16]. Shaked and Bruckstein [26] give a complete analysis of pruning methods and explain their drawbacks.

In the first category, the smoothing of boundaries may cause some undesirable effects on the skeleton, e.g., skeleton shift, and may violate the topological properties, e.g., add new branches to the skeleton [29]. In particular, this category has significant problems in distinguishing between noise and low frequency shape information on boundaries [11]. The scale-axis-transform [12,13] is a different kind of smoothing method that uses a noisy pre-computed medial axis. In 2D, it grows the original shape by linearly scaling the radii of the medial disks and then removing branches of the medial axis that are covered by the larger disks. It preserves the main features of the shape and it reduces low frequency boundary noise. However, it also simplifies the topology of the shape, i.e., it succeeds in creating a new simpler skeleton representation of the shape by simplifying the topology of the shape (e.g., shape holes are occluded when the shape is grown). As a consequence, the skeleton may be shifted far from the medial axis of the original shape. This behavior of the scale-axis-transform may be desirable for some applications, notably level-of-detail tasks, but does not meet our goals.

Liu et al. [16] extend the original idea of the medial axis transform and add a notion of the center of a shape in their extended grassfire transform. They introduce a novel significance criteria for pruning the extended medial axis for possible applications in shape alignment and shape matching. Their experiments show excellent results for shape alignment and matching. However, their pruning technique does not preserve connectivity of the extended medial axis and their solution is not topology preserving [16].

Bai et al. [11] introduce a new concept similar to our solution. They pre-process the shape boundary using a discrete curve evolution to prune the final skeleton. We approximate the shape with a simpler method (i.e., piecewise curve approximation) and introduce a more flexible pruning technique that is also more stable for noisy shapes and under shape rotations. Their solution as ours is capable of conserving the topological properties without shifting the medial axis of the discrete shape or shortening branches. However, they rely on "a priori" knowledge of the shape to prune it, i.e., the number of sides of the polygon to approximate the original shape is required as input parameter to their DCE method (see Fig. 5). Shen et al. [19] introduce a different significance measure for skeleton pruning without boundary approximation, that outperforms the DCE algorithm. The algorithm does not rely on "a priori" knowledge but, its time complexity is higher than our solution and the authors report that the algorithm takes around 7 seconds on average to extract a pruned skeleton from $[512 \times 512]$ images [19] from the same MPEG-7 dataset. Our solution extracts the pruned skeleton of the 1400 images with resolutions from $[256 \times 256]$ to $[950 \times 800]$ in 0.1 s on average (see Section 4). Images with a resolution of $[512 \times 512]$ take about 0.12 s on average.

The main problem in pruning unwanted skeletal points is selecting a good removal threshold. The removal of a point on the medial axis needs to be based only on local information of the surrounding medial axis points [11]. The global information of the shape is usually discarded during skeletonization causing topological changes and shortening of the branches [26]. Discerning noise in the data of a digital skeleton is still an open problem. Hesselink and Roerdink [10] introduce three pruning techniques with their integer medial axis: constant, linear, and square-root pruning, but all of them may disconnect the skeleton and may shorten branches (see Fig. 2).

3. Algorithm overview

Our proposed algorithm has four major steps which we detail in the following: contour approximation, skeleton computation, pruning and skeleton vectorization.

Our solution shares the concept of Bai et al. [11] that every skeleton point is the center of a maximal circle whose boundary tangent points belong to different "sides" of the shape. Skeleton points are pruned if they have all their closest boundary points lie on the same contour segment. Every point in the pruned skeleton remains linked to a boundary point that is tangential to its maximal circle [1] after pruning.

For the first step of our solution, we propose to use a boundary partition that does not require to know the number of vertices a

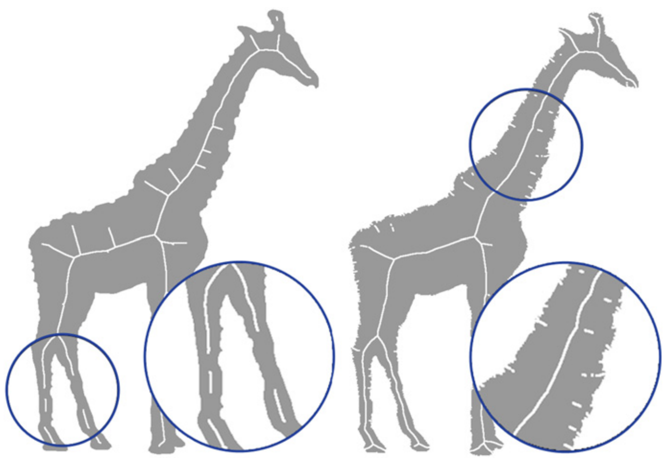

Fig. 2. Distance and angular pruning, both are sensitive to input parameter selection creating disconnected skeletons.

priori. This is in contrast to the discrete curve evolution (DCE) introduced by Bai et al. [11] that requires one to specify the number of vertices of the polygonal approximation beforehand. Intuitively, we want to separate our shape's boundary into different "sides" which we will use in the next step to prune skeleton points associated to only one "side".

The second step of our algorithm computes the integer medial axis transform [10] and starts to prune the skeleton using the contour approximation from the first step. As in [11], there is a skeleton branch at every partition point that needs to be removed after the second step of our algorithm (see Fig. 9). We introduce a novel significance measure based on Blum's medial axis definition of maximal circles, where instead of analyzing isolated skeleton points we check the leaf branches for removal. Leaf branches of the skeleton end at a boundary pixel. This second pruning step handles the elimination of branches that while they have boundary points on different contour approximations because of the knot selection, they are still on the same "side" of the shape. In contrast, Bai et al. use a significance measure calculated with their DCE. Finally, we create a vector representation of the skeleton points using the contour approximation from the first step of the algorithm.

3.1. Contour approximation

For the algorithm's first step, we approximate the binary shape by a piecewise curve. Piecewise curve approximations have been used in the computer graphics industry for a long time. Many methods trying to approximate a set of points by a piecewise curve have been published, e.g., [32–34]. The most challenging problem for these methods is the knot placement to create an optimal approximation. Some of these algorithms may select the knot placement automatically as the methods of Denison et al. [35] and of Plass and Stone [32]. Because we are interested in a real-time algorithm, we need to take into consideration the trade-off between optimal knot placement versus time/space complexity of the contour approximation method. Therefore, we select a simple split-and-merge fitting strategy for knot placement following the algorithm by Ramer [36] as the curve approximation step. Ramer's algorithm splits the approximating curve at points of maximal error which tend to be extremal points. While choosing extremal points may result in suboptimal approximations, it often helps us to make good pruning decisions in later steps of our algorithm. Fig. 3 shows an example where the additional knot required by Ramer's algorithm over optimal node placement produces a better skeleton after pruning.

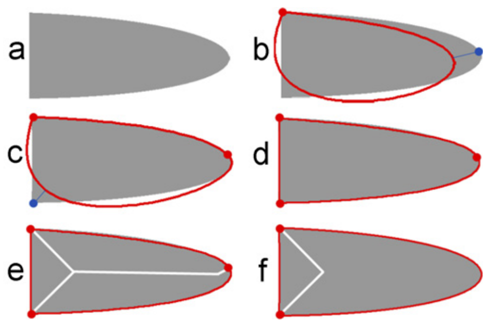

Fig. 3. Example of significant branch removal because of optimal knot placement. Original shape (a) along with the contour approximation steps in (b) to (d) with thresholds $T=10$, $S=1.2$. The suboptimal knots result in a better contour segmentation for final pruning as shown in (e) than the optimal knot placement in (f).

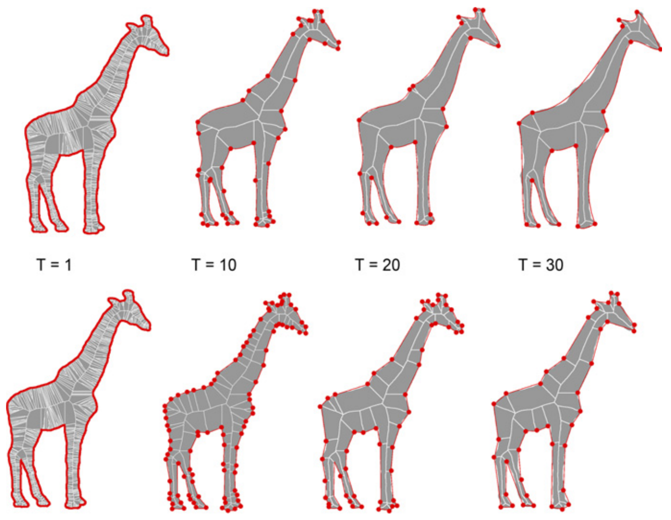

Fig. 4. The effect of different selection of T while pruning the IMA without the final pruning (i.e., the value s is selected as 0). First row shows the results using cubic boundary approximation, second row results are generated with linear approximation.

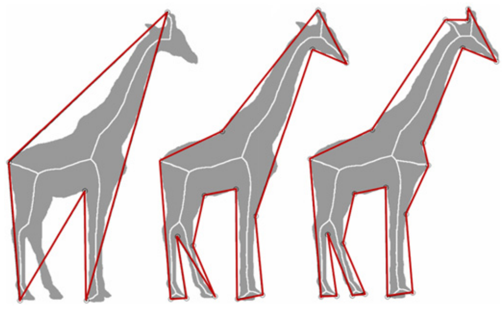

Fig. 5. Pruning using Discrete Curve Evolution DCE. Left: 5 verices, Middle: 15 vertices, Right: 20 vertices.

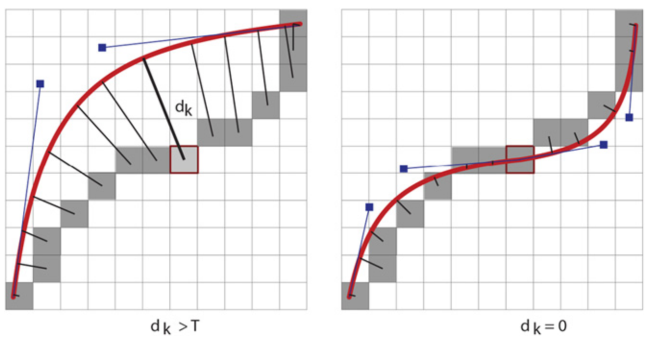

Fig. 6. Piecewise contour approximation example. When we approximate the contour formed by n boundary points, we split the approximating curve (here, a cubic Bezier curve) if the maximum distance from the center of the pixel to the approximating curve is bigger than T threshold selection.

The approximation uses an input threshold T as the maximum distance error allowed between the fitted data and the approximated curve. The piecewise approximation uses the contours of the binary shape to generate a simpler and accurate representation. The fitting strategy can work with any curve but we demonstrate the use of line segments as the simplest curve approximation and cubic Bezier curves with least square fitting. Both, line segments and cubic Bezier curves lead to similar outcomes (see Fig. 4), although cubic Bezier curves require less knots for the same approximation error. The threshold value T determines how closely the approximation fits the original outline. Low values of T remove only higher frequency "noise" from the skeleton without affecting significant parts of the shape while higher values will increase the number of branches removed but without creating a discontinuous skeleton. Our pruning does not shift the skeleton away from the location of medial axis.

Given a shape contour with boundary pixels $C=\{p_1,p_2,\ldots,p_n\}$, we want to find a subset of those pixels and use them as end points of curve sections (e.g., lines or cubics) that approximate the boundary. The shape boundary may be formed by several contours, then the same fitting approach is applied for each of the contours. We select an input threshold error T that specifies the maximum allowed distance between the original points and the fitted curve. Initially, we take the start and end point of the contour as our initial knots and approximate them by a curve, e.g., a line segment with end points p_1 and p_n. Then we measure the minimum distance d_k of the center of each pixel to the fitted curve and look for the point p_k that is farthest from the curve with maximum distance d_k. If the distance from p_k to the curve exceeds the T value, we split the segment at p_k into two new segments p_1,p_k and p_k,p_n fit them using the same curve type (see Fig. 6). This process is repeated for each segment until the T constraint is no longer violated. This produces an ordered set of knots that approximates the shape with a limit error T. The T threshold represents the level-of-detail of the approximation. We empirically selected a threshold of $T=25$ in our experiments. The bigger the error threshold T, the less detail of the shape is taken into consideration and more likely to compute less knots. The smaller T the closest will be the piecewise approximation to the original data (see Fig. 4). A value of $T<0.5$ will make the piecewise curve approximation pixel accurate (i.e., knots will be the original pixel list).

For a cubic parametric curve approximation using Bezier curves we set the initial and last position at two consecutive knot points. Then, using the least square fitting technique we can approximate the best middle control points using the original points between the consecutive knots. This cubic approximation does not consider a continuity constraint on the first derivative of two consecutive segments but reduces the number of required segments. The knot set is typically smaller with cubics than using linear segments for the same T. Cubics have the advantage of fitting a larger amount of points with a single curve section but more computation is required than with the linear approximation.

The least-squares Bezier fitting solves

$$S=\sum_{i=1}^{n}[p_i-q(t_i)]^2 \tag{1}$$

with n data points p_i and the parametric Bezier curve $q(t_i)$. We can write Eq. (1) as

$$S=\sum_{i=1}^{n}\begin{pmatrix} p_i-(1-t_i)^3P_0+3t_i(1-t_i)^2P_1 \\ +3t_i^2(1-t_i)P_2+t_i^3P_3 \end{pmatrix}^2$$

where P_0 and P_3 are the first and last control points; the remaining control points P_1 and P_2 can be determined by setting $\delta S/\delta P_1=0$ and $\delta S/\delta P_2=0$. Solving for P_1 and P_2 gives

$$P_1=(A_2C_1-A_1C_2)/(A_1A_2-A_{12}A_{12})$$

and

$$P_2=(A_1C_2-A_{12}C_1)/(A_1A_2-A_{12}A_{12})$$

where

$$A_1=9\sum_{i=1}^{n}t_i^2(1-t_i)^4,\quad A_2=9\sum_{i=1}^{n}t_i^4(1-t_i)^2$$

$$A_{12}=9\sum_{i=1}^{n}t_i^3(1-t_i)^3$$

and

$$C_1=\sum_{i=1}^{n}3t_i(1-t_i)^2[p_i-(1-t_i)^3P_0-t_i^3P_3]$$

$$C_2=\sum_{i=1}^{n}3t_i^2(1-t_i)[p_i-(1-t_i)^3P_0-t_i^3P_3]$$

After the set of knots is computed, we assign the contour points between two consecutive knots $\{p_k\ldots,p_{k+1}\}$ to the same curve. We also assign an unique identifier for each curve segment. All contours of the shape; exterior and interior (i.e., holes) are processed in the same manner.

3.2. Modified integer medial axis

For the second step, we compute the skeleton by means of the integer medial axis (IMA) without any of the pruning considered by Hesselink et al. [10]. In finding the skeleton we select pixels as skeleton points if they have two equidistant points belonging to two different boundary curves instead of simply two different equidistant points on the boundary as in the original IMA. Our goal is to select skeleton pixels that have equidistant boundary points on two "different" sides of the shape. This criterion will considerably reduce the number of skeleton points detected by the IMA but still produce a connected skeleton (see Section 3.5). Unfortunately, this criterion does not remove all the undesired skeleton points but leaves whole branches that should be removed in place. Branches that end at the intersection of two consecutive curves (see Fig. 7) will be part of the skeleton after this modified IMA. Thus, we need to take extra care of these branches in the pruning step of our algorithm.

Listing 1. Program fragment for filtering the IMA skeleton using our pruning condition. The map C returns the curve associated to a boundary point and map ft2 returns the associated closest boundary point

```
procedure_compare(x,y)
xf:=ft2[x]; yf:=ft2[y];
C1:=C[xf]; C2:=C[yf];
if $\|xf-yf\|^2>1$ and C1 !=C2 then
  crit:=inprod(xf-yf, xf+yf-x-y)
  if crit >0 then skel[x]:=SKEL
  endif
if crit <0 then skel[y]:=SKEL
  endif
endif
```

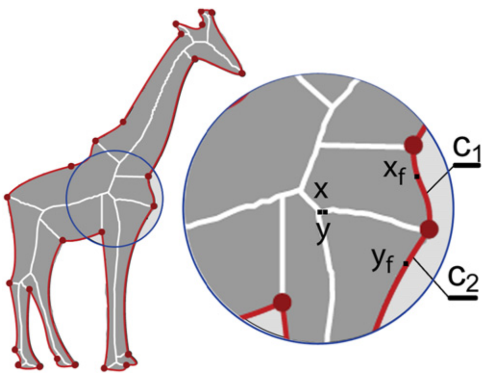

Fig. 7. A Skeleton point X is selected if it has a neighbor point Y with a closest boundary point that belong to a different curves (C_1 vs. C_2) at the same distance d, see Listing 1.

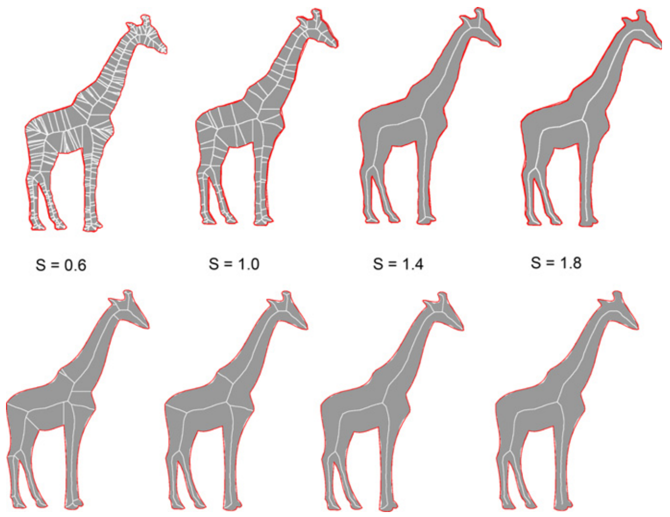

Fig. 8. First row shows the effect of selection of s values for a cubic approximation with fixed value $T=1$, and second row for a cubic approximation with fixed value of $T=25$.

3.3. Final Pruning

For the third step, we need to compute all of the end points and branch points of the skeleton output of the previous IMA step. Assuming a skeleton is one pixel width [10] and always located at a pixel, then those pixels on the skeleton with more than two neighboring pixels are branch points while end points have only one neighboring skeleton pixel. During branch removal, we consider all end points of a branch and their respective branch point belonging to same branch. Our pruning criterion is as follows: Consider a branch $\{b_p,\ldots,e_p\}$ with branch point b_p and end point e_p such that the pixel chain formed by the skeleton points $\{b_p,\ldots,ep\}$ do not contain other branch points or end points except b_p and e_p. Let f be the closest boundary point of b_p as extracted from the feature transform computed as a preprocessing step of the IMA. We mark a branch $\{b_p,\ldots,e_p\}$ for removal if the point e_p is inside of the circle with center b_p and a radius equals to the Euclidean distance $|f-b_p|_2$. We can make this pruning rule stronger if we introduce an input scale factor s for the radius and mark branches for removal if they are inside the scaled circle centered at the branch point (see Eq. (2) and Fig. 9), i.e.,

$$|e_p-b_p|_2\le s*|f-b_p|_2. \quad (2)$$

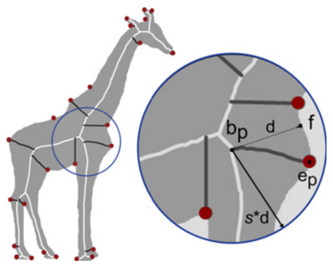

Fig. 9. The unwanted branches (shown dark) introduced in the second step of the algorithm are pruned using Eq. (2). The dark branch to end point e_p is removed by the test at branch point b_p.

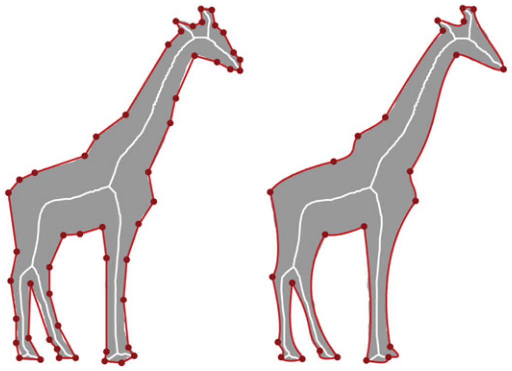

Fig. 10. Final results of the proposed algorithm using linear and cubic approximation contours.

After analyzing all branches for removal, we remove those pixels that belong to the marked branches of the skeleton except for the branch points. We re-iterate this process until no more branch removals are possible. Note that after removing the pixel chain of a branch, we can update the status of the branch points and reduce the computation of searching over the full skeleton set for branch and end points. The final output of the algorithm is a clean skeleton representation (see Fig. 10).

The scale factor s targets the branches that end at the approximation knots created while pruning the IMA. The selection of s will affect the final outcome of the algorithm, where larger values will generally remove more unwanted branches (see Fig. 8).

3.4. Skeleton vectorization

In the final step, we vectorize the skeleton points. Each branch $\{b_p, \ldots, e_p\}$ or $\{b_p, \ldots, b_p\}$ is processed separately and approximated with a piecewise cubic Bezier curve as explained in Section 3.1. We apply an automatic adaptive threshold using the feature transform stored with the IMA from Section 3.2 during the skeleton curve approximation. While processing a branch, we need to check the distance e from every skeleton point x to the approximated curve. If the Euclidean distance e_{max} between the pixel with maximum distance to the approximated curve is bigger than the distance between the point x and its corresponding boundary point x_f stored in the feature transform or if it is bigger than threshold T from the boundary approximation of Section 3.1, we split at x (see Fig. 11). Fig. 12 shows the binary shape input of our algorithm and the final vector representation of the skeleton.

3.5. Correctness and complexity

For showing the correctness of our algorithm we first note that our skeleton output is a subset of the IMA skeleton pixels.

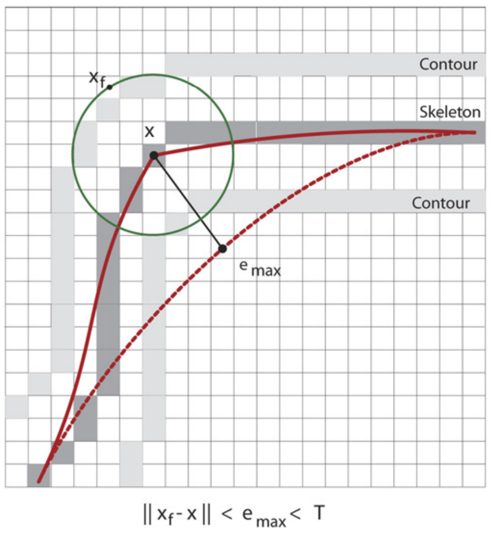

Fig. 11. Skeleton vectorization using the feature transform to automatically adapt the threshold of the curve approximation. The point x has the maximum distance e_{max} to the approximated curve at the point x_f. We split the curve at x if e_{max} is bigger than $\|x_f - x\|$ or if it is bigger than T from Section 3.1.

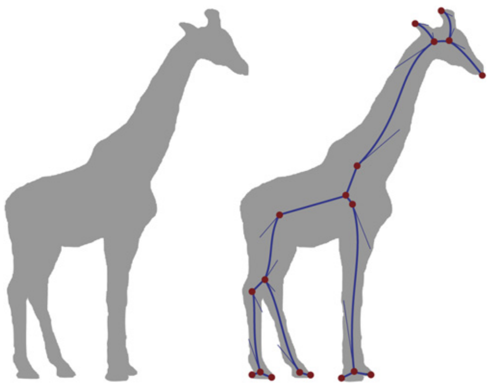

Fig. 12. Input: 2D binary image. Output: Vector representation of the skeleton. Only 19 end points plus 19 control points are necessary to represent 4352 skeleton points using a piecewise cubic approximation.

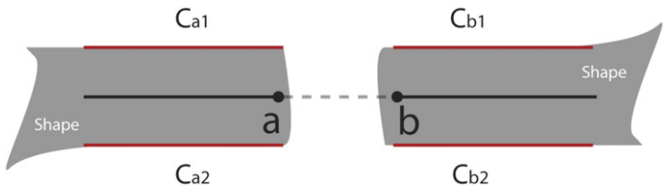

Fig. 13. Pruning the IMA based on contour approximation does not disconnect a skeleton branch. For the skeleton points $\{a, \ldots, b\}$ to be pruned, the boundary contour between C_{a1} and C_{b1} and between C_{a2} and C_{b2} would have to be approximated with the same curve which is impossible.

This proves that every selected pixel is in fact on the medial axis of the discrete shape [11]. Additionally, we need to show that the pruning solution does not disconnect the final skeleton. Let us assume to the contrary that there is a branch that will become disconnected by pruning points $\{a, \ldots, b\}$ from the skeleton. Let the two boundary curves C_{a1} and C_{a2} be linked to skeleton point a, and the curves C_{b1} and C_{b2} be linked to skeleton point b (see Fig. 13). Points $\{a, \ldots, b\}$ will be pruned if and only if the curve connecting C_{a1} and C_{b1} is the same that connects C_{a2} and C_{b2}. This requires that curve C_{a1} and C_{a2} or C_{b1} and C_{b2} are the same. If this is true, then the chains ending at a and b would also be pruned, correspondingly because they would have only one contour segment linked to them. But this contradicts our assumption of a disconnected skeleton. The connectivity of the skeleton is never broken in the final pruning step because only complete pixel chains between an end point and a branch point are removed. We note however that our algorithm may remove complete branches that can be considered significant depending on the boundary approximation.

The boundary approximation step of the algorithm has time complexity of $O(kn+N)$, where k is the number of knots detected during the contour approximation, n is the contour length and N the image size. Computing the contour of the shape is $O(N)$ [22] and the contour approximation has $O(kn)$ with a worst case of $O(n^2)$ for $k=n$ when $T=0$ and all contour points are selected as knots. Assigning the contour pixels to a curve is $O(n)$. The IMA is computed in $O(N)$ [10]. The third step is $O(n)$ because each branch $\{b_p, \ldots, e_p\}$ is only checked once. The vectorization of the skeleton is $0(kn)$. Finally, the total time complexity is therefore $O(kn+N)$. A space complexity of $O(N)$ can be easily achieved.

4. Experimental results

We implemented three skeletonization algorithms for the comparison of our algorithms with the three categories: a thinning approach [15], a Voronoi-based algorithm [22], and a feature transform [10]. Also, five pruning techniques were implemented. Three pruning techniques suggested by Hesselink and Roerdink [10]: constant pruning (DP), linear pruning (LP), and square-root pruning (SRP). A discrete variation of the scale-axis-transform [12,13] and DCE pruning [11] were also implemented. The skeletonization and pruning solutions were tested over the 1400 image MPEG-7 CE Shape-1 Part B dataset; a dataset commonly used for shape analysis [37,38] (e.g., shape matching, shape retrieval, shape skeletonization). The dataset is structured in 20 shape samples of 70 different classes (e.g., bats, planes, beetles, etc.). We account in our tests for running time, accuracy of the skeleton, rotation invariance, shape reconstruction from the skeleton, connectivity of skeleton, and ease of parameter selection. The image resolution of the complete dataset varies from $[256 \times 256]$ to $[950 \times 800]$.

Pruning is a necessary step for the Voronoi-based and IMA algorithms. The thinning approach is more stable and it creates cleaner representations without input parameters or any pruning, i.e., it is fully automatic. However, the skeleton computed from the thinning solution is only an approximation of the medial axis and not the correct medial axis as in the other solutions. Also, the thinning approach is sensitive to noisy boundaries and shape rotations, introducing several unwanted branches for the same shape.

The selection of a correct pruning threshold for distance and angular pruning that is needed for the Voronoi-based algorithm and the IMA is problematic, different threshold values disconnect the shape, e.g., when the leg width of the giraffe in Fig. 5b is less than the selected threshold. In the case of angular pruning with the IMA, it introduced "noise" in areas closer to the boundaries (see Fig. 2). Also, the parameters for the pruning need to change for different shapes. In contrast to our solution where the selection of input parameters is quite stable and we used the same input parameters ($T=25$, $s=1.2$) for the complete dataset. We find the results of our approach also more visually appealing than with the other approaches including the automatic thinning solution (see Figs. 23 and 24).

To compare the running time of the implemented techniques we recorded the computation times of each algorithm. Voronoi-based and thinning solutions were quite slow compared to the other solutions (see Table 1). The IMA is the fastest solution (i.e., IMA + {CP,LR,SRP} are grouped into one category because of their similar results in speed and accuracy), followed very closely by our proposed algorithm using linear and cubic contour approximations.

Shape reconstruction was not possible from the respective outputs of the thinning algorithm and the scale-axis-transform. Based on the skeleton calculated with our proposed algorithm, reconstructing of the original shape is possible with only small error for the complete dataset using the same input parameters ($T=25$, $S=1.2$). For example, the reconstructed shape from the skeleton in Fig. 15 showed 98% overlap with the original shape. The shape can also be fairly accurately reconstructed from the skeleton calculated with the Voronoi-based algorithm, with the IMA and with the DCE but these three algorithms needed different input parameters for different shapes.

In general, all the algorithms were sensitive to noisy boundaries but our solution outperformed the other implemented algorithms on the same image set (see Fig. 16). To complete the comparison, we measured how stable the implementations were under image rotations. For each image in the dataset we first computed their skeleton, then we rotated the original images in steps of 35°, 45°, 70°, 85°, 110°, and 135°, using nearest-neighbor, bilinear, and bicubic interpolation. Each image was rotated around its center. We computed the skeleton of the rotated image using the same parameters as for the skeleton before rotation and compared the two skeletons. We calculated the percentage of matching pixels between the two skeletons after also rotating the pixels of the skeleton of the original image. Besides the

Table 1
Running time comparison.

Algorithm	Running time (95% confidence interval) (s)
IMA + {CP,LP,SRP}	0.08 ± 0.01
Linear	0.09 ± 0.004
Cubic	0.10 ± 0.009
Scale axis transform	0.17 ± 0.013
Thinning	0.35 ± 0.018
Voronoi	0.68 ± 0.021
DCE	5.13 ± 0.518

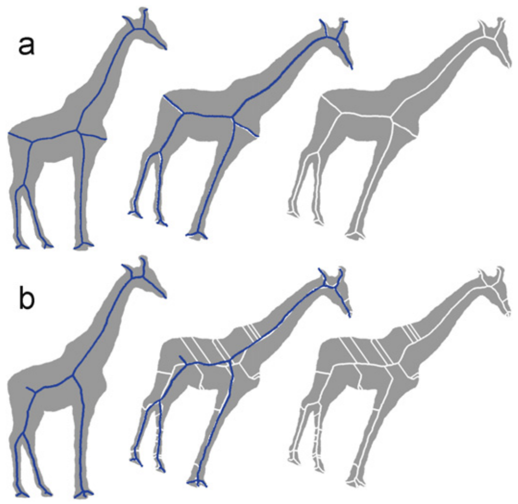

Fig. 14. Example of the performance of DCE (a) and thinning (b) under image rotation. The black skeleton is computed in original image, then, the original image is rotated around the image center (using nearest-neighbor and bilinear interpolation) and the white skeleton is computed. The black skeleton is rotated using the same interpolation method and superimposed on the white skeleton to check for stability. The input parameters of the methods remained unchanged.

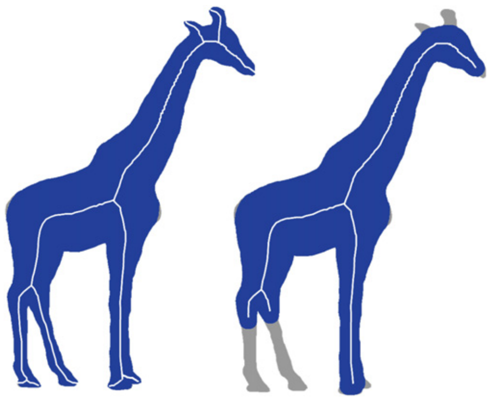
Fig. 15. Example of shape reconstruction using the proposed solution (left) and IMA + constant pruning (right).

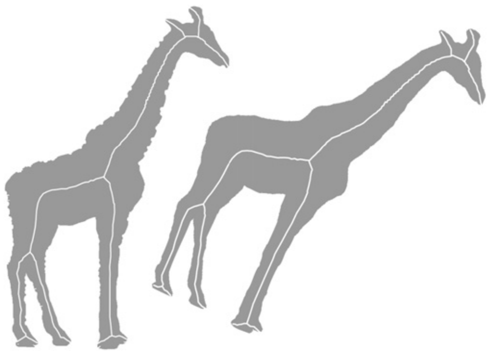
Fig. 16. Output of the proposed algorithm with the same noisy and rotated example. Note the stability of the skeleton detection.

percentage of matching pixels, we also calculated how many pixels were inserted to the medial axis because of the image rotation and how many disappeared from the original skeleton representation (see Fig. 14).

The best matching percentage was obtained by our proposed algorithm using linear and cubic boundary approximations with 95% overlap, followed by the IMA, the scale-axis-transform, DCE and the thinning approach (see Tables 2 and 3). The interpolation method had little influence on the results and the scores were quite similar for nearest-neighbor, bilinear and bicubic interpolation (see also Figs. 18–21).

Finally, we compared the stability of the skeleton of our proposed algorithm against DCE when input parameters are varied. The parameter choice for DCE is specific to each shape (see Fig. 17). Our proposed algorithm allows us to use the same parameters for the complete dataset, yielding excellent results for each of the 70 classes available including for noisy shapes (see Fig. 22). The results obtained by DCE needs parameter tuning in search of an ''optimal'' skeleton representation. In general, the

Table 2
Invariance to image rotations. The matching score gives the percentage of overlapping skeleton pixels detected in both, the original image and in the rotated version.

Algorithm	Matching score (95% confidence interval)
Cubic	0.95 ± 0.002
Linear	0.95 ± 0.004
IMA + {CP,LP,SRP}	0.93 ± 0.005
Scale axis transform	0.90 ± 0.003
DCE	0.78 ± 0.003
Voronoi	0.71 ± 0.009
Thinning	0.49 ± 0.01

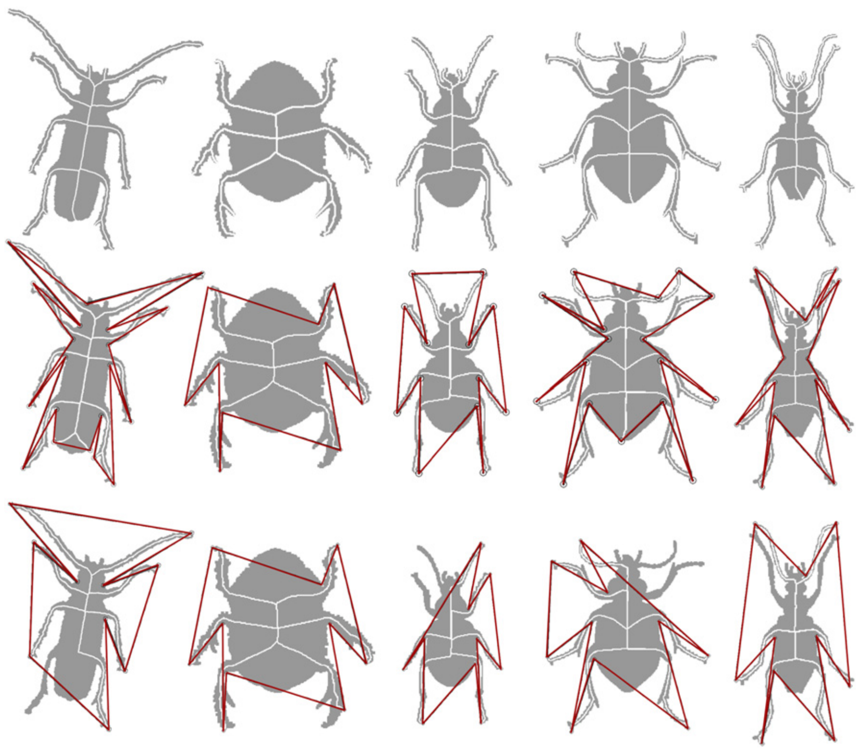
Fig. 17. Parameter choice comparison between our solution and DCE, output examples are from the beetle class of the MPEG-7 dataset. First row is the output of our algorithm using the same configuration parameters ($T=25$, $S=1.2$). The number of cubic segments detected in the first step of our algorithm is 40, 21, 19, 32, and 27 respectively. Second row is the DCE using ''optimal'' user selected input parameters (i.e., number of sides to approximate the original shape, from left to right: 20, 10, 15, 17, 14). Third row is the DCE using constant number of sides equals to 10.

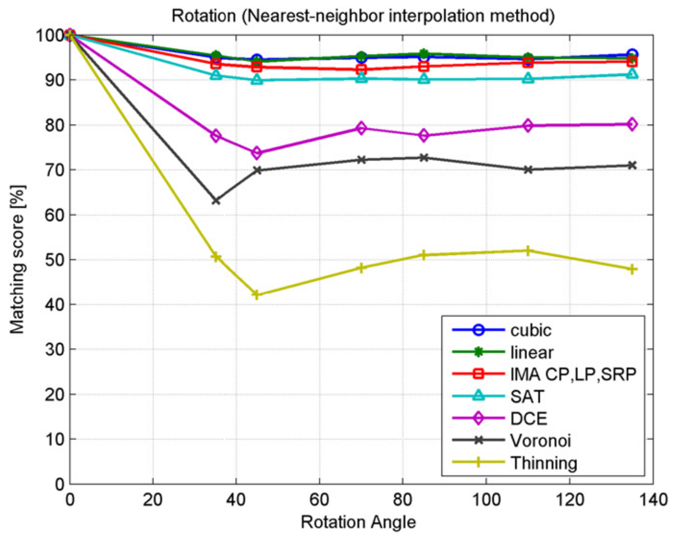

Fig. 18. Image rotation test for angles of 35, 45, 70, 85, 110, and 135 degrees using nearest-neighbor interpolation.

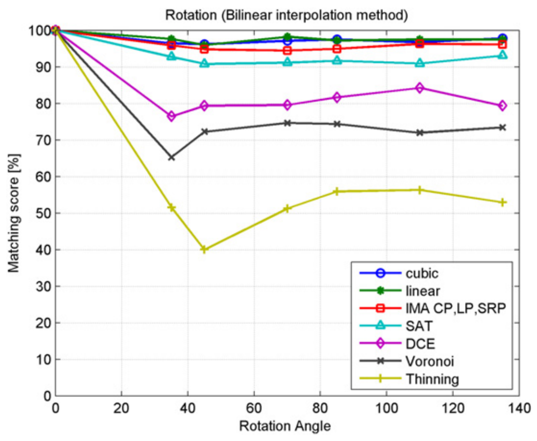

Fig. 19. Image rotation test for angles of 35, 45, 70, 85, 110, and 135 degrees using bilinear interpolation.

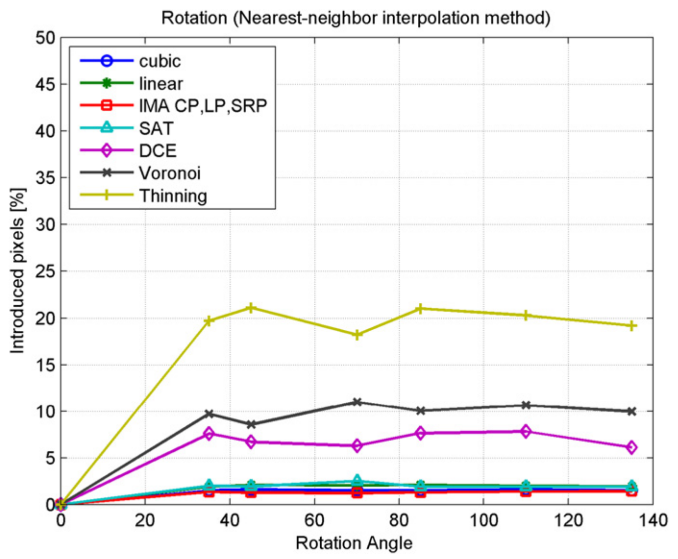

Fig. 20. Percentage of skeleton pixels introduced after the skeletonization of the rotated image (i.e., skeleton pixels that were not detected in the original image but appeared in the rotated version.

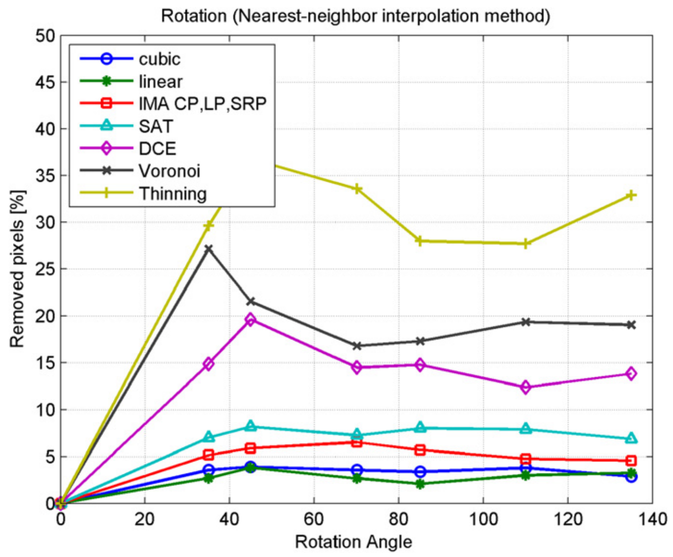

Fig. 21. Percentage of skeleton pixels not present after the skeletonization of the rotated image (i.e., skeleton pixels that were detected in the original image but not in the rotated version).

Table 3
Invariance to image rotations. Inserted/removed pixel score is the percentage of pixels introduced/removed in the skeletonization by the rotation.

Algorithm	Inserted pixels (95% confidence interval)	Removed pixels (95% confidence interval)
IMA + {CP,LP,SRP}	0.013 ± 0.008	0.057 ± 0.007
Cubic	0.015 ± 0.002	0.035 ± 0.002
Linear	0.02 ± 0.0003	0.030 ± 0.003
Scale axis transform	0.02 ± 0.00005	0.08 ± 0.00004
DCE	0.02 ± 0.0006	0.08 ± 0.0002
Voronoi	0.09 ± 0.0012	0.10 ± 0.0015
Thinning	0.19 ± 0.0006	0.32 ± 0.02

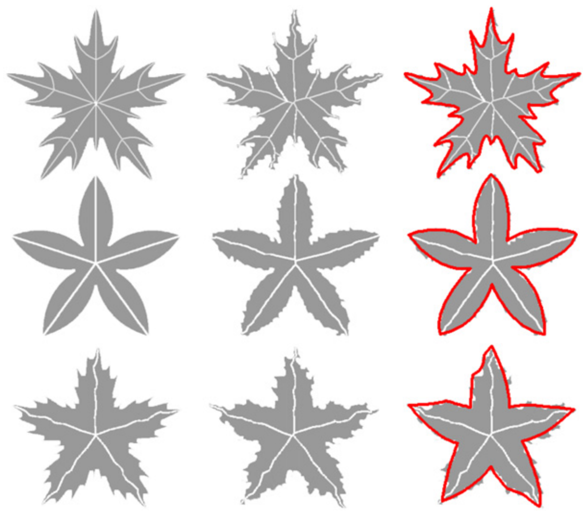

Fig. 22. Output of our algorithm for regular and noisy images of same image class in the MPEG-7 dataset. Last is the same as middle column but with the contour approximation displayed. Parameters are unchanged from other experiments with $T=25$ and $S=1.2$.

outcome of DCE with user supervision for each sample is similar to our results but parameter selection has a big influence on the DCE outcome. It is usually not possible to find a choice that is optimal for all the samples in one class see, e.g., the beetle shape

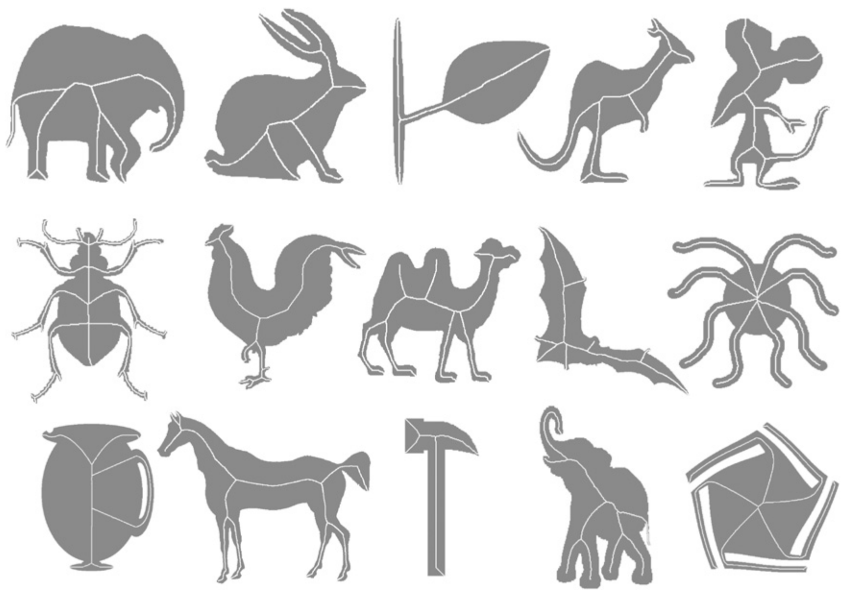

Fig. 23. More output examples from proposed algorithm. The input parameter selection was the same for the whole dataset.

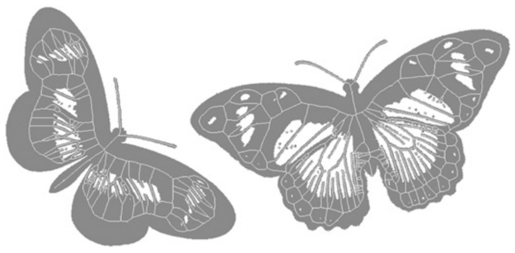

Fig. 24. Some more examples of our proposed algorithm of shapes with holes using the same parameter selection.

in the last column of Fig. 17 where one of the beetle legs has been missed. Parameter tuning for the DCE is also a time consuming process.

5. Conclusions

We presented a new pruning algorithm that removes excessive branches from skeletons of noisy 2D shapes. The algorithm uses a simple contour approximation and a novel branch removal technique. The combination of our two pruning steps proved to be more robust and flexible than using the DCE-based algorithm. Because our algorithm is robust to noise and rigid transformations and it produces consistent high-quality results with the same input parameters, it can be used in semi-automatic or even automatic applications. The broad range of possible applications is also helped by the algorithm's speed and relative ease of implementation. Because our algorithm produces a connected skeleton that is a subset of the integer medial axis of the noisy shape, the shape can be reconstructed from the skeleton using the feature transform. It should also be noted that the integer medial axis can be found efficiently in 3D and we would like to investigate an extension of our algorithm to 3D in future work.

Acknowledgment

We gratefully acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Blum H. A transformation for extracting new descriptors of shape. In: Wathen-Dunn W, editor. Models for the perception of speech and visual form. Cambridge: MIT Press; 1967. p. 362–80.
[2] Siddiqi K, Pizer S. (Eds.), Medial representations: mathematics, algorithms and applications, 1st ed. New York: Springer-Verlag Inc.; 2008.
[3] de Floriani L, Spagnuolo M. (Eds.), Shape analysis and structuring. New York: Springer-Verlag Inc.; 2007.
[4] Eede MV, Mancrini D, Telea A, Sminchisescu C, Dickinson S, Canonical skeletons for shape matching. In: Proceedings of the ICPR, vol. 2; 1997. p. 66–9.
[5] Siddiqi K, Shokoufandeh A, Dickenson SJ, Zucker SW. Shock graphs and shape matching. Int J Comput Vision 1999;35(1):13–32.
[6] Zeng J, Lakaemper R, Yang X, Li X. 2d shape decomposition based on combined skeleton-boundary features. In: Proceedings of the 4th international symposium on advances on advances in visual computing (ISVC), part II; 2008. p. 682–91.
[7] Kégl B, Krzyżak A. Piecewise linear skeletonization using principal curves. IEEE Trans PAMI 2002;24(1):59–73.
[8] Sebastian TB, Kimia BB. Curves vs. skeletons in object recognition. Signal Process 2005;85:247–63.
[9] Moayer B, Fu K. A syntactic approach to fingerprint pattern recognition. Pattern Recognition 1975;7(1–2):1–23.
[10] Hesselink WH, Roerdink JB. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. IEEE Trans PAMI 2008;30(12):2204–17.
[11] Bai X, Latecki L, Liu W. Skeleton pruning by contour partitioning with discrete curve evolution. IEEE Trans PAMI 2007;29(3):449–62.
[12] Giesen J, Miklos B, Pauly M, Wormser C. The scale axis transform. In: Proceedings of the SCG; 2009. p. 106–15.
[13] Miklos B, Giesen J, Pauly M. Discrete scale axis representations for 3d geometry. ACM Trans Graphics 2010;29 101:1–10.
[14] Latecki LJ, nam Li Q, Bai X, yu Liu W. Skeletonization using SSM of the distance transform. In: Proceedings of the ICIP, vol. 5; 2007. p. 349–52.
[15] Lam L, Seong-Whan L, Suen CY. Thinning methodologies—a comprehensive survey, IEEE Trans PAMI 1992;14(9):869-85.
[16] Liu L, Chambers EW, Letscher D, Ju T. Extended grassfire transform on medial axes of 2d shapes. Comput Aided Des 2011;43(11):1496–505.
[17] Arcelli C, di Baja GS. Euclidean skeleton via centre-of-maximal-disc extraction. Image Vision Comput 1993;11(3):163–73.
[18] Di Baja GS, Thiel E. Skeletonization algorithm running on path-based distance maps. Image Vision Comput 1996;14(1):47–57.
[19] Shen W, Bai X, Hu R, Wang H, Latecki LJ. Skeleton growing and pruning with bending potential ratio. Pattern Recognition 2011:196–209.
[20] Zhou R, Quek C, Ng G. A novel single-pass thinning algorithm and an effective set of performing criteria. Pattern Recognition Lett 1995;16(12):1267–75.
[21] She F, Chen R, Gao W, Hodgson P, Kong L, Hong H. Improved 3d thinning algorithms for skeleton extraction. In: Proceedings of the DICTA'09, vol. 1; 2009. p. 14–8.
[22] Haralick RM, Shapiro LG. Computer and robot vision, vol. 1. Addison-Wesley; 1992.
[23] Martínez J, Vigo M, Pla-Garcia N, Ayala D. Skeleton computation of an image using a geometric approach. In: Lensch HPA, Seipel S, editors, EG 2010—short papers; 2010. p. 13–6.

[24] Orrite-Urunuela C, Rincon JMd, Herrero-Jaraba JE, Rogez G. 2D silhouette and 3D skeletal models for human detection and tracking. In: Proceedings of the ICPR, 2004. p. 244–7.
[25] Ogniewicz R, Ilg M. Voronoi skeletons: theory and applications. In: Proceedings of the IEEE CVPR; 1992. p. 63–9.
[26] Shaked D, Bruckstein AM. Pruning medial axes. Comput Vision Image Understanding 1998;69(2):156–69.
[27] Ogniewicz RL, Kübler O. Hierarchic Voronoi skeletons. Pattern Recognition 1995;28:343–59.
[28] Mayya N, Rajan V. Voronoi diagrams of polygons: a framework for shape representation. J Math Imaging Vision 1994:638–43.
[29] Ge Y, Fitzpatrick JM. On the generation of skeletons from discrete Euclidean distance maps. IEEE Trans PAMI 1996;18:1055–66.
[30] Mokhtarian F, Mackworth AK. A theory of multiscale, curvature-based shape representation for planar curves. IEEE Trans PAMI 1992;14:789–805.
[31] Pizer SM, Oliver WR, Bloomberg SH. Hierarchical shape description via the multiresolution symmetric axis transform. IEEE Trans PAMI 1987;9:505–11.
[32] Plass M, Stone M. Curve-fitting with piecewise parametric cubics. SIGGRAPH Comput Graphics 1983;17:229–39.
[33] Horst J, Beichel I. A simple algorithm for efficient piecewise linear approximation of space curves. In: Proceedings of the ICIP, vol. 2; 1997. p. 744–7.
[34] Shao L, Zhou H. Curve fitting with Bezier cubics. Graph Models Image Process 1996;58:223–32.
[35] Denison DGT, Mallick BK, Smith AFM. Automatic Bayesian curve fitting. J R Stat Soc Ser B (Stat Methodol) 1998;60(2):333–50.
[36] Ramer U. An iterative procedure for the polygonal approximation of plane curves. Comput Graphics Image Process 1972;1(3):244–56.
[37] Latecki LJ, Lakämper R, Eckhardt U. Shape descriptors for non-rigid shapes with a single closed contour. In: Proceedings of the IEEE CVPR; 2000. p. 424–9.
[38] Alajlan N, Rube IE, Kamel MS, Freeman G. Shape retrieval using triangle-area representation and dynamic space warping. Pattern Recognition 2007;40(7):1911–20.