Contents lists available at ScienceDirect

Computers & Graphics

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

Technical Section

Spline-based medial axis transform representation of binary images[☆]

Jieying Wang[a,*], Jiří Kosinka[a], Alexandru Telea[b]

[a] Bernoulli Institute, University of Groningen, AG Groningen 9747, the Netherlands
[b] Department of Information and Computing Sciences, Utrecht University, CC Utrecht 3584, the Netherlands

ARTICLE INFO

Article history:
Received 16 December 2020
Revised 21 April 2021
Accepted 30 May 2021
Available online 9 June 2021

Keywords:
Medial axis
B-spline
Shape representation

ABSTRACT

Medial axes are well-known descriptors used for representing, manipulating, and compressing binary images. In this paper, we present a full pipeline for computing a stable and accurate piece-wise B-spline representation of Medial Axis Transforms (MATs) of binary images. A comprehensive evaluation on a benchmark shows that our method, called Spline-based Medial Axis Transform (SMAT), achieves very high compression ratios while keeping quality high. Compared with the regular MAT representation, the SMAT yields a much higher compression ratio at the cost of a slightly lower image quality. We illustrate our approach on a multi-scale SMAT representation, generating super-resolution images, and free-form binary image deformation.

1. Introduction

Binary image encoding plays a key role in applications such as image analysis, matching, and retrieval. It is also important for the *compression* of images and videos [1]. The technologies for binary image encoding can be divided into three classes, namely contour-, bitmap-, and intrinsic methods.

Contour-based encoding represents binary images by a closed contour around their boundary via chain coding [2] or geometrical approximations [3], as detailed further in [4,5]. Approximation methods reconstruct the contour in a lossy manner from a set of representative vertices via polygons or splines, and enable manipulation and deformation [6–10]. *Bitmap techniques* encode the pixels of an image as belonging to the shape (foreground) or outside it (background). These include the modified READ method [11], based on run-length encoding, and context-based arithmetic encoding [12], an efficient entropy coding scheme, which was adopted by the MPEG-4 standards [13].

Intrinsic image encoding considers the interior of a shape rather than its boundary. A key representative of this class uses the shape's medial axis transform (MAT) to encode the shape interior [14,15]. This allows a flexible trade-off between approximation error and required storage [15]. Recently, Zhu et al. [16] followed earlier work that models MATs with multiple cubic B-splines [17] to *automatically* compute a compact spline representation of the MAT of a 2D binary shape. Representing MATs with splines requires fewer (control) points to store than contour-based encoding, which can help image compression. However, this method does not offer a separate control of the MAT simplification and spline approximation.

In this paper, we propose an alternative approach to Zhu et al. [16] that extracts pixel-based MATs directly from binary images and represents them with splines. Our Spline-based MAT (SMAT for short) has the following features:

- Generality: SMAT can directly treat any pixel (binary) image, without requiring the extraction of a densely-sampled vector-representation of the shape contour.
- Algorithm advancement: We propose a more refined spline-fitting scheme, which includes adaptive B-spline fitting and a merge-split algorithm.
- Computational scalability: We inherit the real-time performance of the underlying GPU-based MAT extraction, allowing for high-throughput image processing applications.
- Evaluation: We show that SMAT effectively represents binary images with high accuracy and high compression ratio by measuring five quality metrics.
- Applications: We demonstrate the potential of SMAT by applications in super-resolution image generation, multiscale MAT representation, and free-form image deformation.

The remainder of the paper is organized as follows. Section 2 presents related work in skeletonization, medial axis computation, and B-spline modelling. Section 3 details

[☆] This article was recommended for publication by Dr D. Bommes.
[*] Corresponding author.
E-mail addresses: jieying.wang@rug.nl (J. Wang), j.kosinka@rug.nl (J. Kosinka), a.c.telea@uu.nl (A. Telea).

https://doi.org/10.1016/j.cag.2021.05.012

Table 1
MAT computation methods as a function of the representation of the input I and its MAT S_I, with our method (SMAT) indicated.

		Rep. of ∂I	
		raster	vector
Rep. of S_I	raster	distance field, thinning	distance field
	vector	SMAT	Voronoi methods

the construction of SMAT. Section 4 evaluates SMAT on various images and against five quality metrics. Section 5 presents three applications of SMAT. Section 6 discusses our proposal. Finally, Section 7 concludes the paper.

2. Related work

We structure related work into the computation of MATs (Section 2.1), medial stability and accuracy (Section 2.2), and spline representation models (Section 2.3).

2.1. Medial axis computation

As an intrinsic shape representation, the notion of medial axes was first introduced by Blum [18,19] defined as the locus of centers of maximal disks contained in a shape. Formally, for a binary image $I \in \mathbb{R}^2$ with boundary ∂I, the distance transform DT_I is defined as

$$DT_I(\mathbf{x} \in I) = \min_{\mathbf{y} \in \partial I} \|\mathbf{x} - \mathbf{y}\|. \tag{1}$$

The medial axis, or skeleton, of I is then defined as

$$S_I = \{\mathbf{x} \in I \mid \exists \mathbf{f}_1, \mathbf{f}_2 \in \partial I, \mathbf{f}_1 \neq \mathbf{f}_2 \colon \|\mathbf{f}_1 - \mathbf{x}\| = \|\mathbf{f}_2 - \mathbf{x}\| = DT_I(\mathbf{x})\}. \tag{2}$$

That is, the medial axis S_I of I is the set of points inside I with at least two different closest points, $\mathbf{f}_1$ and $\mathbf{f}_2$, on the boundary ∂I. The points $\mathbf{f}_i$ are also called *feature* points of the medial point $\mathbf{x}$ [20,21]. The medial axis transform MAT of I is the set $\{\mathbf{x} \in S_I, DT_I(\mathbf{x})\}$ of medial points and their maximal disk radii. The union of these maximal disks exactly reconstructs I, which makes the MAT a *dual* representation of shape. For a full discussion of MATs and their properties, we refer to [22,23].

Analytic solutions of Eq. (2) are very hard to compute, and require analytic descriptions of ∂I, which are in general not available. Hence, one computes the MAT by approximating both the input boundary ∂I, but also the MAT itself. Two such main approximations exist: *Raster* methods represent ∂I and/or S_I on a fixed pixel grid; *vector* methods represent ∂I and/or S_I by a piecewise-continuous description in $\mathbb{R}^2$, *e.g.*, using polylines or higher-order curves. With this model, existing MAT computation methods can be further classified as follows (see also Table 1 and related surveys [23,24]).

Morphological thinning methods [25] represent both the image and MAT on pixel grids. They erode I inwards with constant speed until left with a one-pixel-thin connected structure representing S_I. However, such methods do not in general guarantee that S_I is centered within I, *i.e.*, DT_I can be poorly approximated.

Geometric methods [26,27] find S_I as a subset of the edges of the Voronoi diagram of a polyline representation of ∂I. However, such methods require specific sampling conditions for the points describing ∂I to be met [28]. The method of Zhu et al. [16] also falls in this class. While very accurate and compact in representation, finding a vector representation of ∂I for shapes provided in raster (image) form is not evident.

Finally, *distance field* methods [20,21,29–32] compute a raster representation of DT_I from either raster or vector representations of ∂I, and next find S_I along singularities of DT_I. Most current MAT methods fall in this class, given that DT_I can be estimated exactly and in linear time [20,21,32]. Such methods can be further accelerated on the GPU, yielding real-time MAT computation [33,34].

Our method (SMAT) combines the advantages of distance field methods (it accepts any binary image as input, has real-time performance, and computes DT_I accurately) with those of a vector representation of S_I (built-in smoothness, compact storage). SMAT is, to our knowledge, the first (and thus only) method producing vector representations of the MAT from raster representations of input shapes (Table 1).

2.2. Medial axis stability and accuracy

Medial axes S_I estimated from Eq. (2) are notoriously unstable [24]: Small perturbations along ∂I, created *e.g.* by sampling inherent to both raster and vector representations, introduce many so-called spurious medial branches, which contribute little (or not at all in practice) to the description of I, but considerably complicate S_I. Effort has been invested in simplifying medial axes, by removing (parts of) the spurious branches, to make them stable. However, a simplified S_I cannot *exactly* represent, or encode, I. Hence, accuracy (of representing a shape) and stability (of MAT computation) are related, but competing goals. We classify attempts to improve stability and accuracy into two groups, as follows.

Medial-axis-based methods aim mainly to compute a stable, or regularized, $\tilde{S}_I$ by removing spurious branches from S_I. For this, one can estimate the so-called *importance* $\rho(\mathbf{x})$ of every medial point $\mathbf{x} \in S_I$, as the boundary length between the feature points $\mathbf{f}_1$ and $\mathbf{f}_2$ of $\mathbf{x}$. Only medial points with $\rho(\mathbf{x})$ above a user-given threshold are taken over from S_I into $\tilde{S}_I$. Importance thresholding is simple to implement for both raster [31,32] and vector [26,27] medial representations, delivers connected skeletons, and has an intuitive interpretation: the image $\tilde{I}$ reconstructed from $\tilde{S}_I$ replaces all bumps along ∂I shorter than the threshold by circular arcs, effectively acting like a low-pass noise-boundary filter. The *salience* metric [35] replaces $\rho(\mathbf{x})$ by

$$\sigma(\mathbf{x}) = \rho(\mathbf{x}) / DT_I(\mathbf{x}), \tag{3}$$

which removes only the noise-induced medial branches. This yields reconstructions $\tilde{I}$ where small-scale boundary bumps are smoothed out, but important (salient) corners are kept untouched. Other similar metrics include the angle between feature vectors [21,36–38] and divergence of the distance transform [39]. Such metrics have proven very effective in producing simplified stable skeletons (also for 3D shapes [40]). However, they optimize for accuracy only *implicitly*, given the fact that spurious branches represent only small-scale details along ∂I. Distance-based MATs have been also used to compress grayscale images based on the (simplified) MATs of their threshold-sets [41,42].

Reconstruction-based methods approach the joint stability-accuracy problem by maximizing reconstruction accuracy, *i.e.*, the difference between I and $\tilde{I}$ [43]. This can be estimated using the Hausdorff distance [44] between (sampled representations) of ∂I and $\partial \tilde{I}$, defined as

$$H(I, \tilde{I}) = \max\left\{h(I, \tilde{I}), h(\tilde{I}, I)\right\}, \tag{4}$$

where $h(A, B)$ is the one-sided Hausdorff distance given by

$$h(A, B) = \max_{\mathbf{a} \in \partial A}\left\{\min_{\mathbf{b} \in \partial B} \|\mathbf{a} - \mathbf{b}\|\right\}. \tag{5}$$

The MAT method of Zhu et al. [16] uses this approach to compute the simplified $\tilde{S}_I$ by iteratively removing endpoints from S_I, continuously checking their reconstruction error (Eq. (5)) and stopping when this reaches a user-allowed level. This is computationally expensive. Moreover, while H is a recognized metric for

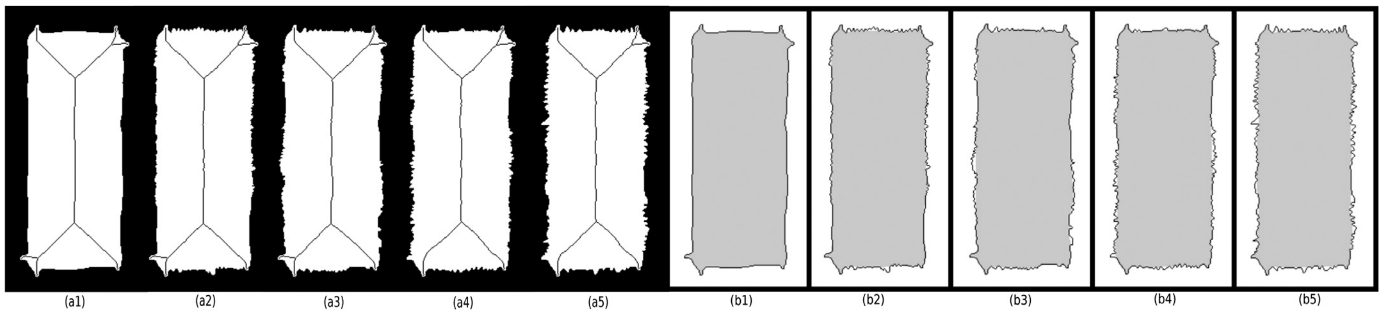

Fig. 1. Five rectangular shapes with different amounts of noise added to the boundary (a1–a5) and the comparisons (b1–b5) between the reconstructions (grey regions) and the original boundaries (black contours) of (a1) and (a5).

Table 2
Accuracy evaluation of the MATs for the images in Fig. 1.

Images	H	$\overline{H}$	Jaccard	MS-SSIM
(a1)	2.2	0.28	0.992	0.991
(a2)	7.0	0.98	0.975	0.959
(a3)	5.0	1.00	0.974	0.955
(a4)	6.3	1.35	0.967	0.936
(a5)	9.0	1.66	0.959	0.921

comparing contours, it is sensitive to noise or outliers [45]. This can prevent regularization: we cannot remove spurious branches from S_I since they will create small changes in $\tilde{I}$ that H highly penalizes. This can be improved by replacing Eq. (4) by a variant using the average operator, *i.e.*,

$$\overline{h}(A,B) = \operatorname*{avg}_{\mathbf{a}\in\partial A}\left\{\min_{\mathbf{b}\in\partial B}\{\|\mathbf{a}-\mathbf{b}\|\}\right\}, \tag{6}$$

leading to the average Hausdorff distance

$$\overline{H}(I,\tilde{I}) = \max\left\{\overline{h}(I,\tilde{I}), \overline{h}(\tilde{I},I)\right\}. \tag{7}$$

Besides considering the similarity of *boundaries*, one can take the overall shapes into account. The Jaccard similarity coefficient [46] achieves this by considering the size of the intersection, normalized by the size of the union, of two shapes

$$Jaccard(I,\tilde{I}) = \frac{|I\cap\tilde{I}|}{|I\cup\tilde{I}|}. \tag{8}$$

The Multi-Scale Structural SIMilarity (MS-SSIM) index [47,48] provides an advanced top-down model of how the human visual system interprets images. Although designed to measure the similarity of grayscale images, it can also be used for binary images. Both Jaccard and MS-SSIM range in [0,1], where 1 indicates the two input shapes are exactly the same, while 0 means the two are completely different.

Fig. 1 (a1–a5) shows five rectangular shapes with randomly added noise on their boundary, and their simplified medial axes for $\sigma_0 = 1.5$. We see that, as the noise increases, the simplified medial axes change little, and are thus quite stable to noise.

Table 2 shows the above four quality metrics H, $\overline{H}$, Jaccard, and MS-SSIM for the reconstructed images. The more noise on the boundary, the larger H and $\overline{H}$, and the lower the Jaccard and MS-SSIM scores. To give an intuitive understanding of these values, Fig. 1 (b1–b5) compares the reconstructed images (gray) with the original boundaries (black) of (a1–a5), respectively. For (b1), the two almost overlap, yielding a very small $\overline{H} = 0.08\%$ of the image diagonal, and Jaccard and MS-SSIM scores above 0.99. The reconstruction in (b5) preserves the salient features of the original image while removing small-scale boundary noise. This still yields a small $\overline{H} = 0.45\%$ of the image diagonal, and high Jaccard and MS-SSIM scores.

2.3. B-spline representation

Storing and manipulating simplified medial axes $\tilde{S}_I$ and their corresponding MATs can benefit from the observation that such structures correspond to piecewise smooth curves (branches) [23]. Following this, Yushkevich et al. [17] first proposed to model the MAT with cubic B-splines. However, their method needs to build a template continuous medial representation model *manually*, which is then manipulated to fit a target shape. Zhu et al. [16] improve this by proposing a fully *automatic* way to represent MATs with B-splines. Yet, their method handles only *vector* representations (of both the shape and its medial axis). In contrast, our method uses *raster* representations for both I and S_I (Section 3), and converts the latter into a vector representation using splines. This (1) makes our method directly applicable to any binary image, without the need to extract a piecewise-continuous contour ∂I with sampling guarantees; and (2) provides vector-based medial representations for *any* raster-based MAT computation method, in contrast to [16], which only works with the Voronoi-based MAT method of [27].

B-splines are a common and preferred way of specifying very smooth curves ($C^{(d-1)}$ continuity for degree d) in computer graphics and geometric design [49]. Given $n+1$ control points (CPs) $\mathbf{p}_0,\dots,\mathbf{p}_n$ and a knot vector $U = [u_0,\dots,u_m]$, the B-spline curve of degree d is defined as [50]

$$\mathbf{c}(u) = \sum_{i=0}^{n} N_{i,d}(u)\mathbf{p}_i. \tag{9}$$

The functions $N_{i,d}(u)$ are the *B-spline basis functions*, defined recursively via

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \\ N_{i,d}(u) &= \tfrac{u-u_i}{u_{i+d}-u_i}N_{i,d-1}(u) + \tfrac{u_{i+d+1}-u}{u_{i+d+1}-u_{i+1}}N_{i+1,d-1}(u). \end{aligned} \tag{10}$$

A B-spline curve given by $n+1$ control points, $m+1$ knots, and degree d must satisfy $m = n+d+1$. The knot vector is either *open* or *periodic*. In this work, we use *open-uniform* knot vectors, given by

$$u_i = \begin{cases} a, & 0 \le i \le d \\ a + \dfrac{i-d}{n+1-d}(b-a), & d < i \le n \\ b, & n < i \le n+d+1 \end{cases}, \tag{11}$$

where a and b are usually set to 0 and 1 respectively. This allows generating a B-spline curve based only on a set of $n+1$ control points and degree d with $0 < d < n+1$.

3. Proposed SMAT representation

We compute our SMAT representation as follows (see also Fig. 2). We start with a binary image I as input. For simplicity of

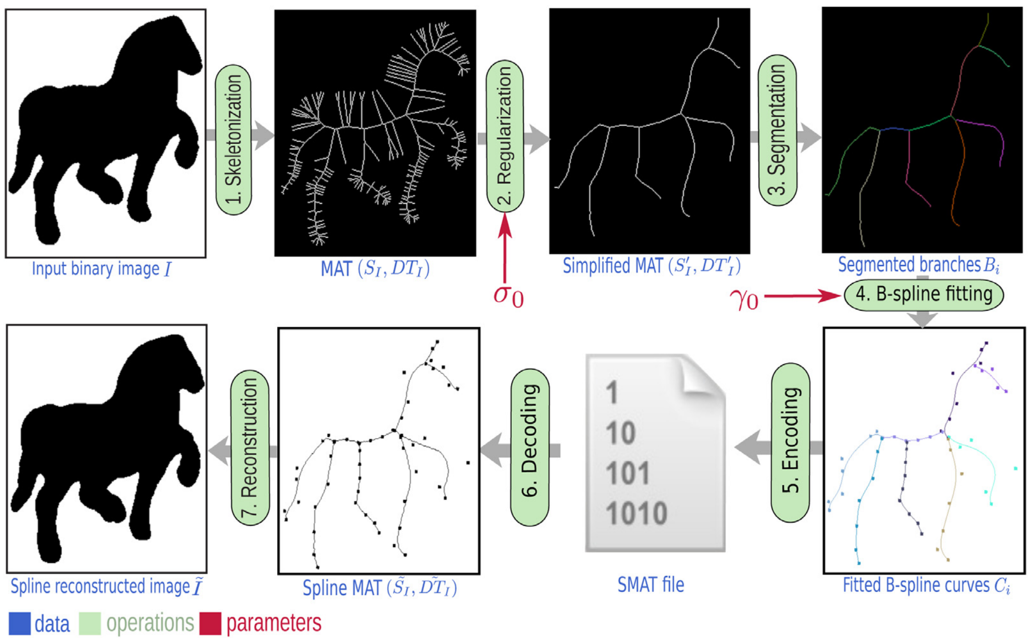

Fig. 2. Pipeline of the proposed SMAT representation.

exposition, we next consider I has a single foreground connected component (black pixels in Fig. 2a); in practice, our implementation handles multiple such components, one at a time. We next compute the full MAT (S_I, DT_I) using the method in [31]. However, any other raster-based MAT computation can be used as well. Next, we compute the simplified medial axis

$$S'_I = \{\mathbf{x} \in S_I \mid \sigma(\mathbf{x}) \geq \sigma_0\}, \tag{12}$$

by upper-thresholding the salience metric σ (Section 2.2) by a user-specified value σ_0. The simplified MAT is given by (S'_I, DT'_I), where DT'_I is the restriction of DT_I to the pixels of S'_I. We use the simplified MAT as input for our SMAT construction. For this, we segment S'_I into separate branches (Section 3.1) and fit them using splines (Section 3.2). Finally, we encode the entire set of spline control points efficiently (Section 3.3). The resulting encoding can be then used to reconstruct an approximation $\tilde{I}$ of the input shape I (Section 3.4).

Algorithm 1: Skeleton segmentation algorithm

```
Input: Simplified MAT $(S'_I, DT'_I)$
Output: Set $\mathbf{B}$ of medial branches
1  Make $S'_I$ 8-connected
2  Find $\mathbf{x}$ = a point in $S'_I$
3  $\mathbf{B} = \varnothing$
4  Trace($\mathbf{x}$, $B$)
5  return B
6
7  Function Trace($\mathbf{y}$, $B$):
8      $B$.push_back($\mathbf{y}$, $DT'_I(\mathbf{y})$)
9      erase $\mathbf{y}$ from $S'_I$
10     if size($\mathbf{y}$.neighbors) = 1 then
11         Trace($\mathbf{y}$.neighbors[0], B)
12     else
13         add $B$ to $\mathbf{B}$
14         for n in y.neighbors do
15             Trace($\mathbf{n}$, $\varnothing$)
```

3.1. Medial axis segmentation

To carry out the *piecewise* B-spline fitting, the simplified medial axis should be segmented into branches (Fig. 2, Step 3). Algorithm 1 outlines the segmentation procedure. First, we clean up S'_I so it is 8-connected. We next characterize medial points by the number of neighbors in S'_I they have, as follows: Branch *endpoints* have a single neighbor; *regular* points have two neighbors; and branch *junctions* have three or more neighbors. We then find an endpoint (or an arbitrary point if no endpoints exist) $\mathbf{x}$ of S'_I and start tracing along the medial axis from there, adding the discovered MAT points $(\mathbf{y}, DT'_I(\mathbf{y}))$ to the current branch B. When arriving at a junction or endpoint, we add B to the branch-set $\mathbf{B}$ and, if at a junction, start tracing new branches from all medial neighbors n of the current point $\mathbf{y}$.

3.2. B-Spline fitting

To each branch $B_i = \{(\mathbf{x} \in S'_I, DT'_I(\mathbf{x}))\}$ found in the branch-set $\mathbf{B} = \{B_i\}$, we fit a B-spline curve C_i using a least-squares algorithm (Fig. 2, step 4). For details of the least-squares fit, we refer to [51]. Given a user-provided fitting error γ_0 between B_i and C_i, we compute the minimal number of needed control points N and spline degree d by an adaptive algorithm (Section 3.2.1). We further minimize the number of needed splines C_i across several branches by a merge-split algorithm (Section 3.2.2).

3.2.1. Adaptive-degree fitting

Although it is common to use quadratic ($d = 2$) or cubic ($d = 3$) B-splines to approximate a set of points, we compute d so as to get the lowest number of control points N needed to reach an error below the user-given threshold γ_0. Computing d follows the constraints $1 \leq d < N$ (see Section 2.3) and $N \geq 2$, where $N = 2$ implies a line segment fit to the branch B. The procedure (Algorithm 2) is a global minimization of N for all possible d values within a maximum number of iterations *MaxIter* (set in practice to 1000) to speed up computations for large branches. The fitting error γ of the spline C to branch B is given by the Hausdorff distance $H(B, C)$ (Eq. (4)) computed over all the pixels $\mathbf{x} \in B$. It is known that the boundary approximation error $H(I', \tilde{I})$ between I' (the reconstruction from the simplified MAT (S'_I, DT'_I)) and $\tilde{I}$ is upper bounded

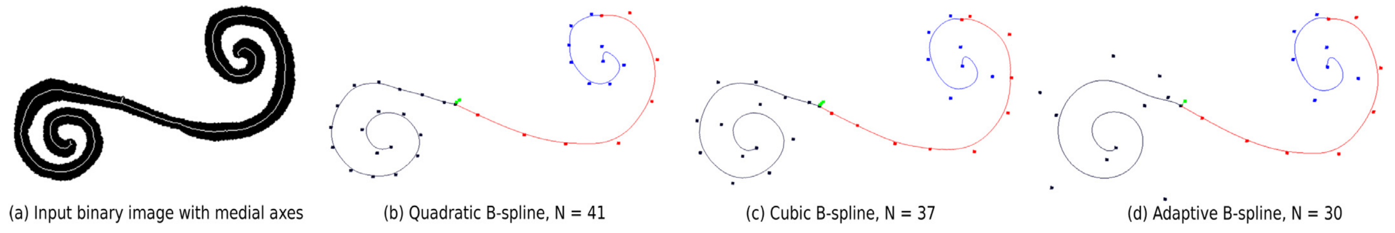
(a) Input binary image with medial axes
(b) Quadratic B-spline, N = 41
(c) Cubic B-spline, N = 37
(d) Adaptive B-spline, N = 30

Fig. 3. Comparison of the adaptive B-spline fitting (d) with quadratic (b) and cubic (c) B-spline fitting for a given image (a) and error $\gamma_0 = 0.5\%$ of the image diagonal. For (b–d), different colors show different branches. Control points are colored like their branches.

Algorithm 2: Adaptive B-spline fitting algorithm.

```
   Input: MAT branch $B$ and user-given maximal error $\gamma_0$
   Output: B-spline curve $C$ fitted to $B$ under error $\gamma_0$
1  Initialization: $N$ = 2; $\gamma_{\min}$ = INFINITY
2  for $i$ from 0 to MaxIter do
3      for $d$ from 1 to $N-1$ do
4          $C$ = LeastSquaresFit($d$, $N$, $B$)
5          $\gamma = H(B, C)$
6          if $\gamma < \gamma_{\min}$ then
7              $\gamma_{\min} = \gamma$; $d_{\min} = d$
8      if $\gamma_{\min} < \gamma_0$ then
9          break
10     else
11         $N$++
12 $C$ = LeastSquaresFit($d_{\min}$, $N$, $B$)
```

Table 3
Number of control points N needed to fit each of the four medial branches for the quadratic, cubic, and adaptive schemes in Fig. 3. Values in brackets give the degree d of each B-spline.

Branch	Quadratic	Cubic	Adaptive
B_1 (blue)	10 (2)	8 (3)	8 (3)
B_2 (red)	9 (2)	11 (3)	9 (2)
B_3 (green)	3 (2)	4 (3)	2 (1)
B_4 (black)	19 (2)	14 (3)	11 (5)
Total	**41**	**37**	**30**

by $\sqrt{2}\gamma_{\max}$ [52], where $\gamma_{\max}$ is the maximal fitting error over all splines C_i. Thus, users can set γ_0 either directly — to control the *medial axis* approximation error — or based on the desired *boundary* approximation error $H(I', \tilde{I})$.

Fig. 3 compares our adaptive B-spline fitting with quadratic and cubic B-spline fitting for a simple shape. Fig. 3(a) shows the image I (black) and its medial axis S_I' (white). We set here $\gamma_0 = 0.5\%$ of the image diagonal. As visible, our adaptive fitting (Fig. 3(d) requires only $N = 30$ control points, whereas quadratic and cubic B-splines require $N = 41$ and $N = 37$ control points, respectively. Table 3 lists the number of control points and degree for the four medial branches of the shape, for each of the above three spline-fitting schemes. For long and curved branches, like B_4 (black), our scheme uses a high degree $d = 5$ to reduce the required N; for short and straight branches, like B_3 (green), our method reduces N by using a lower degree $d = 1$ than the quadratic and cubic schemes. Overall, our adaptive-degree B-spline fitting saves about 20% control points.

3.2.2. Merge-split algorithm

Algorithm 2 describes the adaptive B-spline fitting for a *single* branch. When the medial axis S_I is only slightly simplified, S_I' contains many short branches corresponding, as outlined in Section 2.2, to small-scale details along ∂I. Encoding these requires many control points, so we propose to *merge* these short branches to alleviate this. For each branch-fragment A, we find all fragments

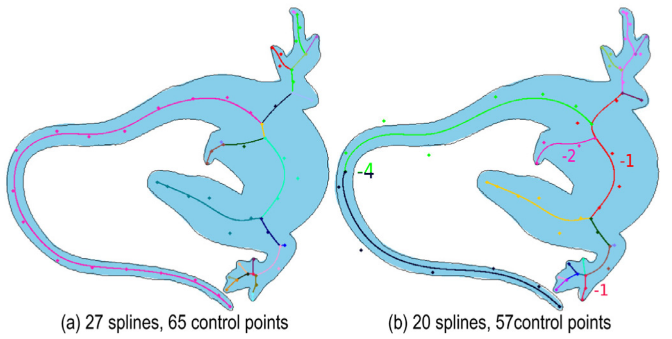
-4
-2
-1
-1
(a) 27 splines, 65 control points
(b) 20 splines, 57control points

Fig. 4. Comparison of SMAT for a lizard shape without merge-split (a) and with this algorithm used (b). Cyan shows the SMAT-based reconstruction of the shape. Branch segments are shown in different colors. The negative numbers in (b) show the number of control points saved per branch by MS.

B_i connected to it, and select the most suitable one B_j to merge with A. The criterion for the merge is

$$j = arg\,min_i \; N_{A+B_i} \mid N_{A+B_j} < N_A + N_{B_j}, \tag{13}$$

where N_{A+B_i} is the number of control points needed to fit the merged branch $A \cup B_i$, and N_A and N_{B_j} are the number of control points required for branch-fragments A and B_j, respectively.

A separate issue is that there may be long and curved branches which are difficult to fit with a B-spline, requiring more than $N_{\max}$ (set to 15 in practice) control points. We address this by *splitting* such branches before fitting. Let $B_i = \{(\mathbf{x}_k \in S_I', DT_I'(\mathbf{x}_k))\}_{k=0}^{n}$ be the branch to be split; then $B_{i1} = \{(\mathbf{x}_k \in S_I', DT_I'(\mathbf{x}_k))\}_{k=0}^{n/2}$ and $B_{i2} = \{(\mathbf{x}_k \in S_I', DT_I'(\mathbf{x}_k))\}_{k=n/2+1}^{n}$ are the two branches obtained by splitting B_i *in half*. We consecutively split long and curved branches in half until all resulting branch-segments can be fitted by Algorithm 2 with splines with fewer than $N_{\max}$ control points, and the fitting error is under the user-specified γ_0.

To illustrate the effect of the merge-split (MS) algorithm, Fig. 4 compares SMAT results for a lizard shape (taken from [16]) when applying the MS algorithm (b) and not applying it (a). We set $\gamma_0 = 0.35\%$ of the diagonal of the image. MS creates a SMAT using only 57 control points instead of the 65 required if one B-spline per MAT branch is used. The negative numbers in Fig. 4 show the drop in control points, per branch, due to MS. The long and curved branch along the tail is split into two segments (black and green). Fewer control points are needed after splitting since the two segments can be separately fit with splines using different degrees. For short and straight branches, like the ones corresponding to a finger of the fore leg, merging them into one branch also decreases the control point count. Additionally, using MS greatly reduces the total number of splines from 27 to 20 used since several connected MAT branches can be approximated by a single spline. More information on the advantages of MS for reducing the information needed to store the SMAT is given in the supplementary material.

3.3. SMAT encoding

Our SMAT representation (Fig. 2, Step 5) consists of a tuple $\mathcal{SM} = (w, h, \{\mathbf{b}_i\})$. Here, w and h represent the width and height, in pixels, of the input image I; and $\mathbf{b}_i = (d_i, \{\mathbf{c}_i^j\})$ represents a B-spline output by the merge-split process (Section 3.2.2), *i.e.*, which fits a merged branch under the user-given error γ_0. Each such B-spline has a degree d_i and control points $\mathbf{c}_i^j$ (both computed by the adaptive procedure in Section 3.2.1. Each control point $\mathbf{c}_i^j = (\mathbf{p}_i^j, DT_I'(\mathbf{p}_i^j)) \in \mathbb{R}^3$ consists of a 2D position $\mathbf{p}_i^j$ and its corresponding DT value.

3.4. Reconstruction

We reconstruct the approximation $\tilde{I}$ of I from the SMAT representation (Section 3.3) as follows. For each spline $\mathbf{b}_i$, the *open-uniform* knot vector can be determined following Eq. (11). The basis functions $N_{i,d}(u)$ are next computed using Eq. (10), followed by generating the B-spline (Eq. (9)). Each such spline is next rasterized on the desired pixel grid, using either the original image resolution (w, h) or, if desired, a higher resolution (see next Section 5.1). For rasterization, we first split each branch $\mathbf{b}_i$ into Bézier (polynomial) segments. This uses knot insertion to ensure that each internal knot has multiplicity equal to d_i, which is done efficiently using the Oslo algorithm [53]. Each Bézier segment is then rasterized using adaptive binary subdivision based on de Casteljau's algorithm [50]. The adaptive subdivision proceeds until the maximum distance of all inner Bernstein-Bézier control points from the line-segment given by the end-points of the current (sub-)segment is below pixel precision, at which point the (sub-)segment is drawn using Bresenham's line-drawing algorithm based on the mentioned end-points.

Once the medial pixels $\mathbf{x}_i$ with DT values DT_i are evaluated this way, we reconstruct $\tilde{I}$ as the union of disks with centers $\mathbf{x}_i$ and radii DT_i, using the efficient procedure for computing this union from [31].

3.5. Implementation

We implemented the MAT simplification (Eq. (12)), Algorithms 1 and 2, the SMAT encoding (Section 3.3), and spline rasterization (Section 3.4) in C++. We compute exact Euclidean MATs and also reconstruct the initial image from a rasterized SMAT using the public CUDA implementation provided at [54]. Our entire method, including source code, datasets, and evaluation scripts, is publicly available [55].

4. Results

4.1. Evaluation methodology

SMAT depends on two parameters (Fig. 2): the salience threshold σ_0, which gives the simplification of the medial axis S_I', and the tolerance γ_0 that tells how accurately B-splines fit medial branches. We evaluate SMAT based on two factors:

Similarity Q of the reconstruction $\tilde{I}$ provided by SMAT (Section 3.4) to the original shape I. Here, Q stands for any of the metrics H, $\overline{H}$, MS-SSIM, and Jaccard (Section 2.2).

Compression ratio CR that measures how more compact $\mathcal{SM}(\tilde{I})$ is as compared to I. We define $CR = |I|/|\mathcal{SM}(\tilde{I})|$. Here, $\mathcal{SM}(\tilde{I})$ is the size (in bytes) of the SMAT storage scheme outlined in Section 3.3. In contrast, $|I|$, *i.e.*, the storage needed for shape I, can be defined in many ways, depending on how I is represented, *e.g.* by chain coding [2], geometrical approximation [3], or bitmap-based encoding [11,12]. We next model $|I|$ as the size (in bytes) of the contour ∂I.

Table 4
Evaluation of SMAT for an animal shape.

Shape	σ_0	γ_0 (%)	Similarity metrics Q				CR	N
			$H(\%)$	$\overline{H}(\%)$	Jaccard	MS-SSIM		
Animal 4368 Bytes (400*235)	0.1	0.2	0.47	0.11	0.981	0.986	2.1	540
		0.4	0.47	0.12	0.979	0.984	3.0	358
		0.6	0.47	0.13	0.978	0.984	3.3	327
		0.8	0.60	0.13	0.978	0.983	3.3	329
	0.5	0.2	0.78	0.23	0.960	0.958	10.2	105
		0.4	0.69	0.18	0.970	0.970	10.6	101
		0.6	0.69	0.22	0.960	0.959	11.3	93
		0.8	0.88	0.20	0.966	0.966	12.6	83
	1.0	0.2	0.88	0.27	0.952	0.946	24.4	43
		0.4	1.2	0.24	0.957	0.953	28.0	36
		0.6	1.1	0.26	0.955	0.951	33.9	33
		0.8	1.2	0.29	0.949	0.942	35.2	27
	1.5	0.2	1.2	0.28	0.950	0.944	24.9	42
		0.4	1.1	0.24	0.958	0.954	28.1	37
		0.6	1.3	0.26	0.953	0.947	33.6	33
		0.8	1.1	0.27	0.952	0.946	36.4	27

Table 5
SMAT benchmark with 30 images of five types.

Type	Description
a	Simple object shapes
b	Regular geometric structures
c	Animal contours
d	Geometric shapes with jagged or irregular edges
e	Shapes with complex contours

Given the above, the total quality of SMAT can be modeled as

$$(Q, CR) = \text{SMAT}(\sigma_0, \gamma_0). \tag{14}$$

To find a good trade-off between Q and CR, we do a grid-search over σ_0 and γ_0. We use $\sigma_0 \in \{0.1, 0.5, 1.0, 1.5\}$ as this range was indicated in the original salience paper [35] as producing MAT simplifications that remove small-scale noise but keep salient details. We use $\gamma_0 \in \{0.002, 0.004, 0.006, 0.008\}$ (percentages of the bounding-box diagonal of I) as these represent tight fits of the SMAT B-splines with the branches of S_I'. Table 4 shows the resulting values for all four similarities Q, the compression ratio CR, and also the total number of control points N in the SMAT, for all values of σ_0 and γ_0, for a binary image also used in [31,39]. Several other examples are available in the supplementary material.

To understand the trends in Table 4, we use next two scatterplots (Figs. 5) showing H *vs* CR and MS-SSIM *vs* CR, respectively. The plots of $\overline{H}$ *vs* CR and Jaccard *vs* CR are similar to Fig. 5(a) and (b), respectively, and are not included for space constraints. In each plot, the sixteen points correspond to combinations of (σ_0, γ_0) values. We encode σ_0 in four base colors (hues), and γ_0 in the size of the bullets. Fig. 5(a) shows a roughly direct correlation of H with CR, while Fig. 5(b) shows an inverse correlation of MS-SSIM with CR, as expected. In general, the larger σ_0, the more simplified S_I', so the lower the MS-SSIM and the higher the H and CR. Overall, the quality has not decreased much, but the CR has been greatly improved. Under a certain σ_0, γ_0 determines how close the B-spline is to the current skeleton. Although there is no very strict trend, in general, with the increase of γ_0, Q tends to decrease while CR increases.

4.2. Joint compression-quality evaluation

We next evaluate SMAT's ability to compress images *and* retain similarity by a benchmark containing 30 images of five different types (Table 5 and Fig. 6), mainly selected from the MPEG-7 benchmark [56]. To optimize both reconstruction similarity Q and CR for

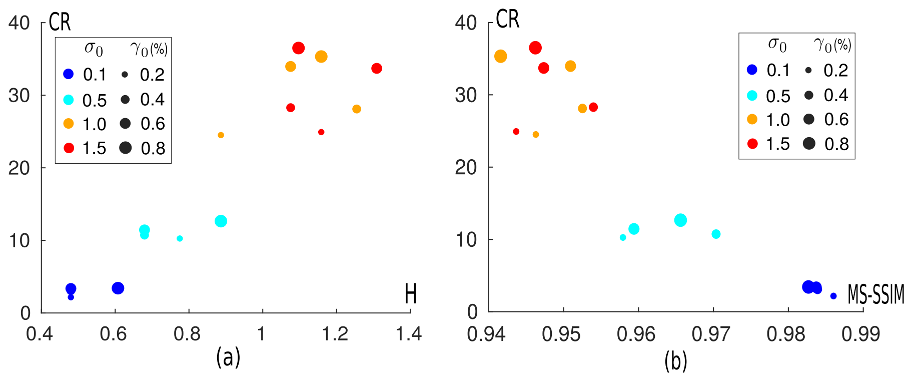

Fig. 5. *H vs CR* (a) and MS-SSIM *vs CR* (b) for the shape in Table 4.

each shape, we next set $Q =$ MS-SSIM, as we argue that, from the set of four considered similarity metrics, MS-SSIM best represents how humans perceive two shapes I and $\tilde{I}$ as being similar. Also, we combine MS-SSIM and CR into a simple joint quality metric

$$Q' = \frac{MS\text{-}SSIM + \overline{CR}}{2}, \tag{15}$$

where $\overline{CR}$ for a given shape is its CR value normalized by the maximal CR over all shapes in the benchmark. This way, both CR and MS-SSIM range in [0,1], so can be combined in Eq. (15). The optimization is the result that maximizes Q' over all 16 studied (γ_0, σ_0) values.

Fig. 6 shows the obtained results. Rows indicate shapes in the five benchmark classes (see Table 5). Cyan shows the reconstruction $\tilde{I}$ with optimal Q'. Black outlines show the boundaries ∂I of the input shapes. In all cases, the reconstruction is visually almost identical to the original shape, also confirmed by the high MS-SSIM values. Compression values CR also are in general quite high, less so for shapes having many small-scale details such as (d6) and (e6). Images (a6), (e1), and (e6) show that SMAT can handle shapes with holes with no problem. Relative boundary length, defined as boundary length $|\partial I|$ divided by the diagonal of the image, is color-coded in the legend in Fig. 6. Fig. 7 summarizes the SMAT performance on the 30 images in Fig. 6 by a scatterplot of MS-SSIM *vs CR*. Colors indicate the relative boundary length just as in Fig. 6. We see a slight inverse correlation of high CR and MS-SSIM with the relative boundary length, which is expected: shorter boundaries have, on average, fewer details, so are easier to encode by SMAT. This is also visible in the fact that complex shapes (types d and e) reach lower CR and MS-SSIM, while simpler shapes (types a and b) reach higher CR and MS-SSIM. We also see the relatively strong effect that even tiny boundary details have on CR: Shapes a1, b1, b4, and b2 are arguably of very similar visual complexity and all have short boundaries (dark blue in Fig. 7). All compress with a very high MS-SSIM > 0.985. However, their CR's vary between 37.2 (a1) and 133 (b2). This is caused by tiny, pixel-size, noise, present *e.g.* along a1's boundary, but largely absent for the other three shapes. To keep such tiny details, we need to keep many branches in the MAT.

4.3. Comparison with compressed MAT representation

Besides comparing SMAT with the ground-truth I, it is interesting to compare it with other methods that provide *compressed* MAT representations. One such method [41] essentially uses the same MAT extraction [31] and simplification [35] as SMAT, but next compresses the pixel-chains in S_I' using delta encoding rather than B-splines as we do. Fig. 8 shows the average MS-SSIM *vs* CR for the five shape types in our benchmark for SMAT and the delta method in [41]. For that method, we define $CR = |I|/|MAT(\tilde{I})|$, where MAT is the size (in bytes) needed to store S_I' with delta encoding for a binary image, which is an efficient way to store 8-connected pixel-paths. As in our case, (Section 4.1), $|I|$ denotes the size (in bytes) needed to store ∂I.

In the figure, the larger the γ_0 (the larger the filled dots), means the larger the approximation error of the spline, so the lower the quality, the higher the CR. When γ_0 equals 0.002, the MS-SSIM score of SMAT is only 0.002 lower than the one of the delta encoding, but SMAT yields CR values 2 up to 6 times higher.

4.4. Comparison with Zhu et al. [16]

Result quality: Fig. 10 compares SMAT with Zhu et al. [16], which, as mentioned earlier, is the most similar method (in aims) to ours. This comparison must however be done carefully. As mentioned (Section 2.3), their input shape boundaries ∂I must be carefully (densely) sampled to capture the boundary topology faithfully [28]. How this is done is not further detailed. Also, they use for reconstruction error Q the *one-sided* Hausdorff distance (Eq. 5) from these *sample* points $P \in \mathbf{P}$ of ∂I to the boundary $\partial\tilde{I}$ of the reconstructed shape. This leads to a smaller distance value than if *all* points of ∂I were considered. Fig. 9 illustrates this. If the (one-sided) Hausdorff distance is defined as $h(I, \tilde{I}) = \max_{\mathbf{x}_i \in \mathbf{P}} \left\{ \min_{\mathbf{x}_j \in \partial\tilde{I}} \|\mathbf{x}_i - \mathbf{x}_j\| \right\}$ as in [16], then some of the distance values (like l in the figure) will be less than (maximally equal to) values obtained when considering *all* points on ∂I (like l' in the figure). In addition, they only considered the one-sided Hausdorff distance $h(I, \tilde{I})$, for reasons deemed as simplicity. The other one-sided Hausdorff distance, $h(\tilde{I}, I) = \max_{\mathbf{x}_j \in \partial\tilde{I}} \left\{ \min_{\mathbf{x}_i \in \mathbf{P}} \|\mathbf{x}_j - \mathbf{x}_i\| \right\}$, would generate a higher value than $h(I, \tilde{I})$ (denoted m in the figure).

In contrast to the above, Fig. 10 (a2–c2) shows the *two-sided H* (Eq. 4), and considers *every* pixel on both boundaries ∂I and $\partial\tilde{I}$. We used here images extracted from the paper [16], since no data or code for that method was available. In this image, ϵ is the one-sided Hausdorff distance used by [16] explained above. The other metrics are ours, explained earlier. Overall, SMAT produces results that are very similar to Zhu et al., but requires 15% fewer control points N. Importantly, H, the double-sided Hausdorff distance we use for SMAT, is always larger than the one-sided Hausdorff distance ϵ used by Zhu et al., by definition – meaning that our test for accurate reconstruction is more stringent than the one in Zhu et al.

Parameters: Zhu et al. offer a single parameter $\hat{\epsilon}$ which controls *both* the MAT simplification and the spline fitting. Small $\hat{\epsilon}$ values, thus, will keep most of the raw MAT branches and also tightly fit B splines to them. Conversely, large $\hat{\epsilon}$ values will simplify the MAT significantly and fit B splines looser. In contrast, we

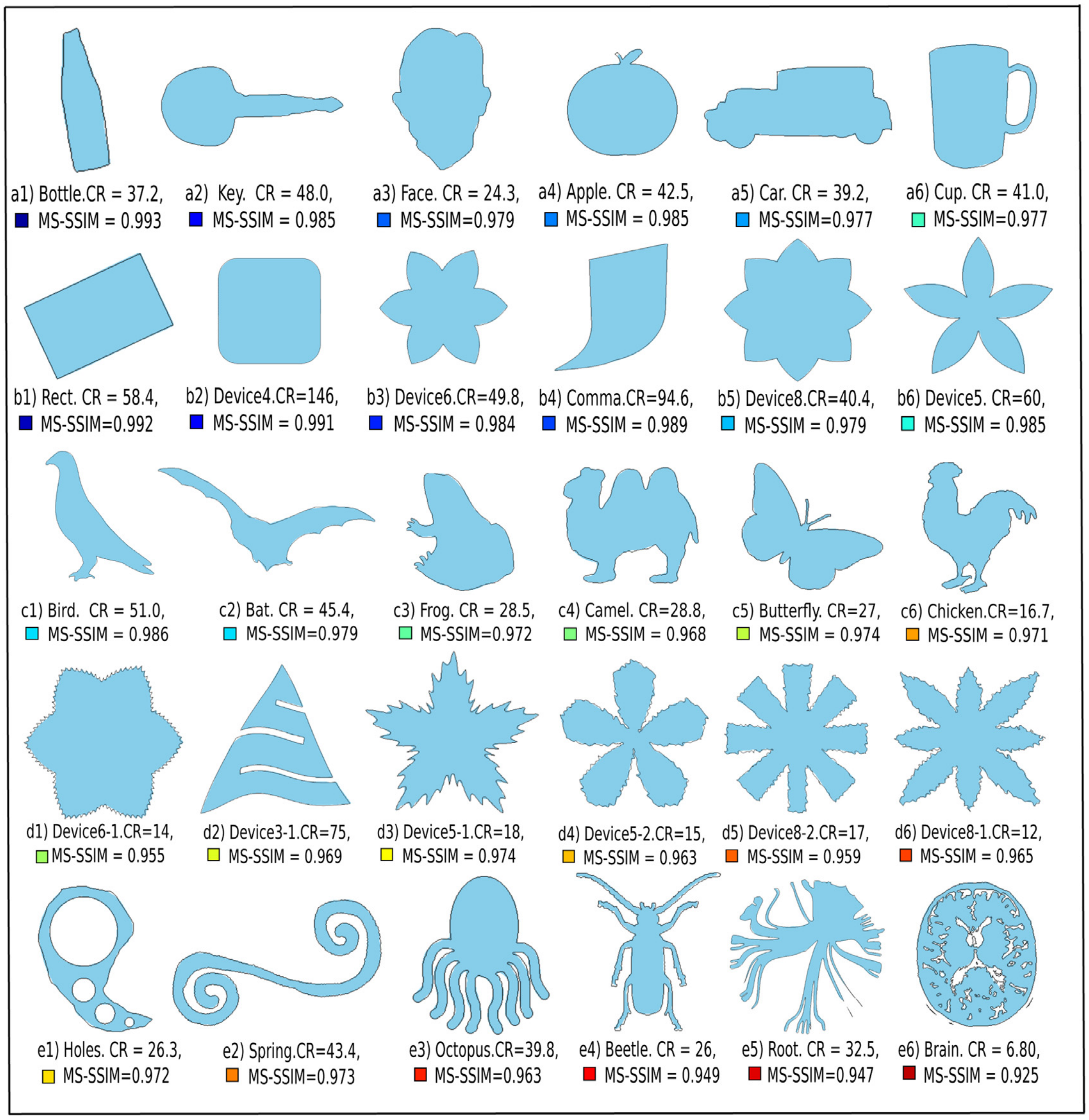
a1) Bottle.CR = 37.2, MS-SSIM = 0.993
a2) Key. CR = 48.0, MS-SSIM = 0.985
a3) Face. CR = 24.3, MS-SSIM=0.979
a4) Apple. CR = 42.5, MS-SSIM = 0.985
a5) Car. CR = 39.2, MS-SSIM=0.977
a6) Cup. CR = 41.0, MS-SSIM=0.977
b1) Rect. CR = 58.4, MS-SSIM=0.992
b2) Device4.CR=146, MS-SSIM = 0.991
b3) Device6.CR=49.8, MS-SSIM = 0.984
b4) Comma.CR=94.6, MS-SSIM = 0.989
b5) Device8.CR=40.4, MS-SSIM = 0.979
b6) Device5. CR=60, MS-SSIM = 0.985
c1) Bird. CR = 51.0, MS-SSIM = 0.986
c2) Bat. CR = 45.4, MS-SSIM=0.979
c3) Frog. CR = 28.5, MS-SSIM=0.972
c4) Camel. CR=28.8, MS-SSIM = 0.968
c5) Butterfly. CR=27, MS-SSIM = 0.974
c6) Chicken.CR=16.7, MS-SSIM = 0.971
d1) Device6-1.CR=14, MS-SSIM = 0.955
d2) Device3-1.CR=75, MS-SSIM = 0.969
d3) Device5-1.CR=18, MS-SSIM = 0.974
d4) Device5-2.CR=15, MS-SSIM = 0.963
d5) Device8-2.CR=17, MS-SSIM = 0.959
d6) Device8-1.CR=12, MS-SSIM = 0.965
e1) Holes. CR = 26.3, MS-SSIM=0.972
e2) Spring.CR=43.4, MS-SSIM=0.973
e3) Octopus.CR=39.8, MS-SSIM=0.963
e4) Beetle. CR = 26, MS-SSIM = 0.949
e5) Root. CR = 32.5, MS-SSIM=0.947
e6) Brain. CR = 6.80, MS-SSIM = 0.925

Fig. 6. Comparison of SMAT reconstructions (blue) with original shapes (black outlines) for 30 images in our benchmark. Colored boxes left of the labels show the relative length of the shape contour (blue is short, red is long). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

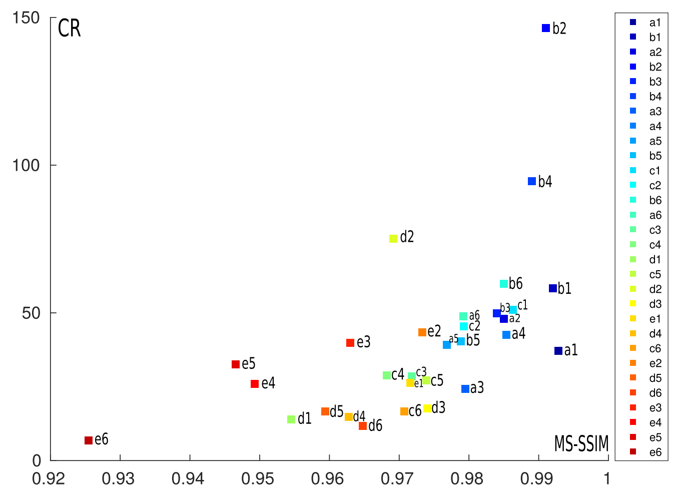

Fig. 7. MS-SSIM *vs* CR of 30 shapes for SMAT representation. Colors encode the relative boundary length just as in Fig. 6.

separate the two concerns: Our parameter σ_0 controls the MAT simplification, thereby allowing us to specify *what* (of the MAT) we next want to approximate. Next, γ_0 controls the spline fitting, *i.e.*, *how well* we want to approximate the simplified MAT. This separation has several advantages: (1) We can *e.g.* decide we want to keep the nearly-complete MAT (low σ_0) but approximate it more loosely (high γ_0), or simplify the MAT significantly (high σ_0) but fit it tightly with splines (low γ_0). (2) We can use any other MAT simplification besides the saliency, simply by replacing the definition of S_I' with any other MAT simplification deemed suitable; the rest of the pipeline stays unchanged. (3) We can compute multi-scale SMATs by computing the full MAT only once and next simplify it using any desired combination of σ_0 and γ_0, at interactive rates (see Section 5.2 for details).

Speed: Zhu et al. only report the cost of computing the final B-splines from the *simplified* MATs. Yet, going from the raw MATs (produced by the Voronoi method [27]) to the simplified MATs takes tens of seconds for a shape of 512^2 pixels. In contrast, our *total* time, from receiving I up to computing $\mathcal{SM}(\tilde{I})$, is only tens of milliseconds (see Fig. 10 a2–f2).

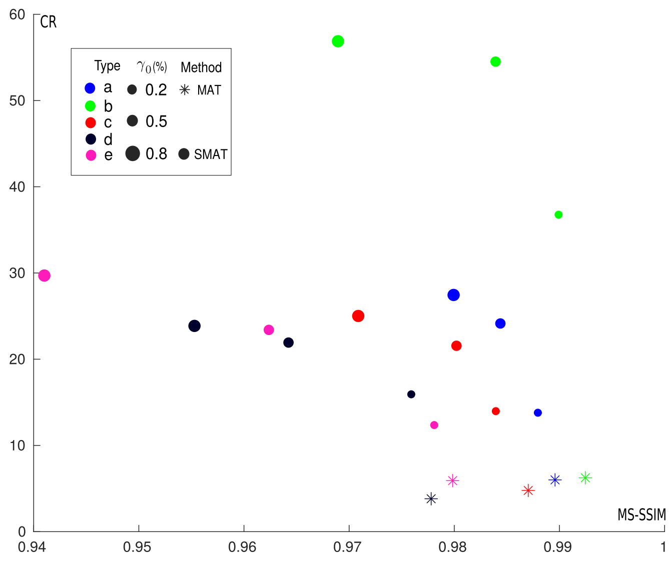

Fig. 8. Average MS-SSIM *vs CR* for the five shape types in our benchmark for delta-encoding MAT [41] (asterisks) and SMAT for three different γ_0 values (filled dots in three different sizes).

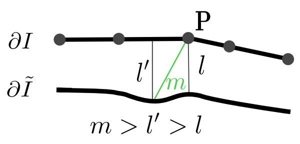

Fig. 9. Analysis of Hausdorff distance used in [16]. P is one of the sample points of the original shape boundary ∂I. $\partial \tilde{I}$ is the boundary of the reconstructed shape.

5. Applications

In addition to the shape or MAT compression (Section 4), SMAT also provides other useful results. We outline here super-resolution medial axes (Section 5.1), multiscale SMAT representations (Section 5.2), and shape manipulation by MAT control points (Section 5.3).

5.1. Super-resolution

As B-splines are smooth, one can *rasterize* them at any resolution in the reconstruction step (Section 3.4). This does not incur any extra storage: the same SMAT representation (Section 3.3) can be used. Fig. 11 shows the effect of increasing the resolution by 10 times on a jagged shape. The top images show the SMAT reconstruction of a shape at its original resolution (200×200 pixels). Bottom images show the SMAT reconstruction, *from the same SM representation*, at a 10 times higher resolution than the input image. As visible, the super-resolution reconstruction removes the discretization artifacts of the original reconstruction while keeping the reconstructed boundary $\partial \tilde{I}$ (line separating black from white in the image) smooth. To our knowledge, this is the only method providing super-resolution raster MATs apart from [57]. In comparison to [57], SMAT is fully generic and simpler, *i.e.*, does not need to use special (image-based) interpolation tricks to create the super-resolution MATs.

5.2. Multiscale SMAT representation

SMAT uses as input the *simplified* MATs S_I' of its input shapes I (Section 3). These simplified MATs depend on the user-provided salience parameter σ_0. In practice, this means that if a user simplified S_I' too much (by setting σ_0 too high), one would have to re-run the entire SMAT pipeline, which is tedious and time consuming. We improve this by proposing a *multiscale* SMAT representation.

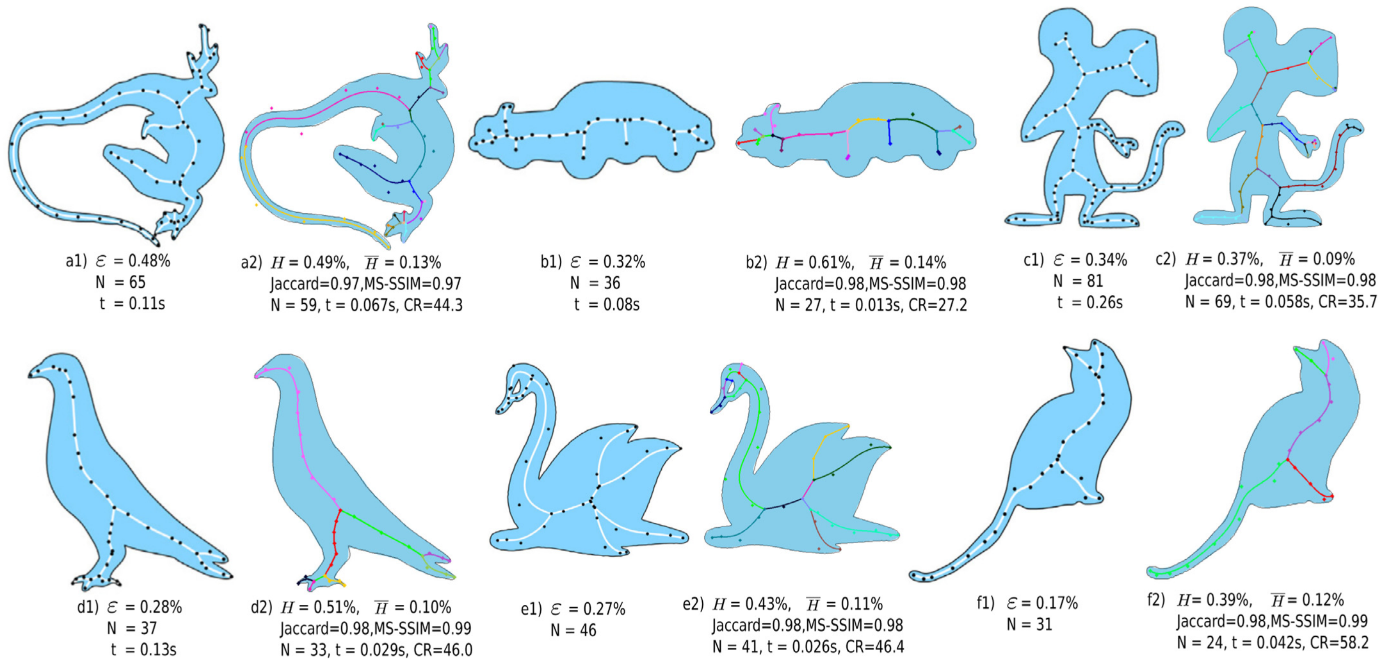
a1) $\mathcal{E}$ = 0.48%
N = 65
t = 0.11s
a2) H = 0.49%, $\overline{H}$ = 0.13%
Jaccard=0.97,MS-SSIM=0.97
N = 59, t = 0.067s, CR=44.3
b1) $\mathcal{E}$ = 0.32%
N = 36
t = 0.08s
b2) H = 0.61%, $\overline{H}$ = 0.14%
Jaccard=0.98,MS-SSIM=0.98
N = 27, t = 0.013s, CR=27.2
c1) $\mathcal{E}$ = 0.34%
N = 81
t = 0.26s
c2) H = 0.37%, $\overline{H}$ = 0.09%
Jaccard=0.98,MS-SSIM=0.98
N = 69, t = 0.058s, CR=35.7
d1) $\mathcal{E}$ = 0.28%
N = 37
t = 0.13s
d2) H = 0.51%, $\overline{H}$ = 0.10%
Jaccard=0.98,MS-SSIM=0.99
N = 33, t = 0.029s, CR=46.0
e1) $\mathcal{E}$ = 0.27%
N = 46
e2) H = 0.43%, $\overline{H}$ = 0.11%
Jaccard=0.98,MS-SSIM=0.98
N = 41, t = 0.026s, CR=46.4
f1) $\mathcal{E}$ = 0.17%
N = 31
f2) H= 0.39%, $\overline{H}$= 0.12%
Jaccard=0.98,MS-SSIM=0.99
N = 24, t = 0.042s, CR=58.2

Fig. 10. Comparison of SMAT (a2–f2) with [16] (a1–f1) for six shapes. Blue shows the reconstructed shapes $\tilde{I}$. Black shows boundaries of the input shapes I. White curves and black dots in (a1–f1) show the B-spline curves and their corresponding control points. In (a2–f2), different colors show different branches and their control points. The errors ε, H and $\overline{H}$ are in percentages of the diagonal of the image. Timings t for (a1–f1) cover only the spline extraction from simplified MATs in [16]. Our timings t (a2–f2) are the end-to-end costs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

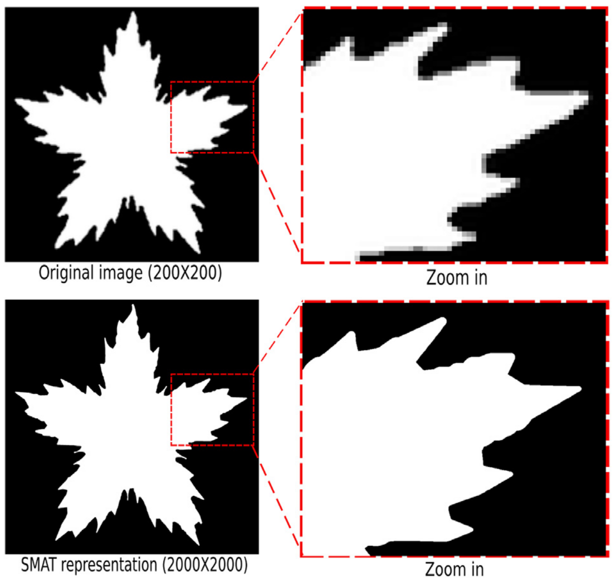

Fig. 11. Super-resolution effect on a shape with jagged edges.

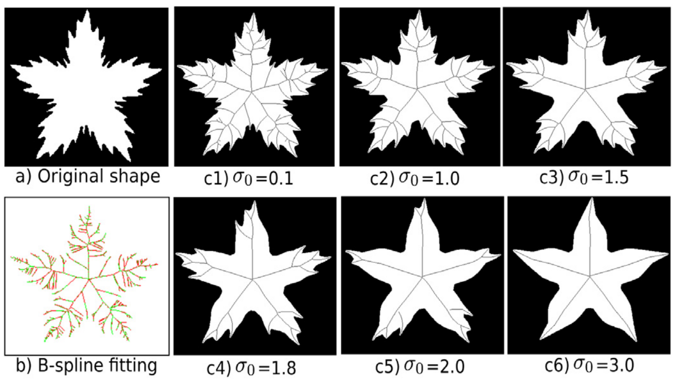

Fig. 12. Multiscale SMAT fitting (b) and the progressively simplified skeletons and the corresponding reconstruction (c1–c6) on a jagged shape (a).

Instead of simplifying the MAT (Fig. 2, Step 2), we encode the *full* MAT together with the importance (ρ) value at each MAT point as a 4D curve-set $(S_I.x, S_I.y, DT_I, \rho)$, using B-splines. As we encode both DT_I and ρ, we can next compute the salience σ using Eq. 3.

A single multiscale SMATs allows generating an entire family of *progressively simplified* MATs, and their corresponding reconstructions, by running only the σ_0 thresholding on the decoded $\tilde{S_I}$ produced by Step 6 in Fig. 2, followed by reconstruction (Step 7). Fig. 12(b) shows the 4D data $(\tilde{S_I}.x, \tilde{S_I}.y, DT_I, \rho)$ in red, fitted by B-splines (green) for the jagged shape in Fig. 11. Videos showing additional results on this experiment are in the supplementary material. Figs. 12(c1–c6) show the gradually-simplified MATs S_I' and corresponding reconstructions $\tilde{I}$ for increasing thresholds σ_0. Multiscale SMAT is cost-effective: obtaining the six simplifications in Fig. 12(c1–c6) is about five times faster than running six single-scale SMATs. Note that producing such multiscale SMATs is not possible with [16]: Indeed, for any change of the error user parameter $\hat{\epsilon}$ proposed there, the entire pipeline of MAT simplification and spline-fitting has to be re-run, which is expensive, as outlined in Section 4.4.

5.3. Shape manipulation

Manipulating 2D shapes is important in applications like character animation [58,59] and image editing [60]. Several methods such as *m-reps* [61], subdivision skeletons [62], and medial surface deformation [63] have used MATs to this end, by changing the shape via changing its medial axis. Still, picking suitable control points to manipulate the MAT is not easy. In contrast, SMAT allows deforming a shape simply by manipulating the SMAT descriptor. Fig. 13 shows five deformations where we keep the human body and horse rump fixed and change the horse legs by adding ($+c$), deleting ($-c$), moving (m) control points and/or increasing the B-spline degrees ($+d, -d$). A few of the manipulations are outlined next. To overlap the two hind legs and vary the curvature of their joint silhouette (d, e) we increased the number of control points for the leg-to-body connection branch (dark blue) and also reduced the number of control points for these legs. A similar edit was done to the purple branch in (f) to overlap the front legs. To get a curled up right front leg (dark red, image (e)), we reduced the degree of its B-spline to 1 so as to better control it to clearly show each leg joint, and next moved its control points as desired.

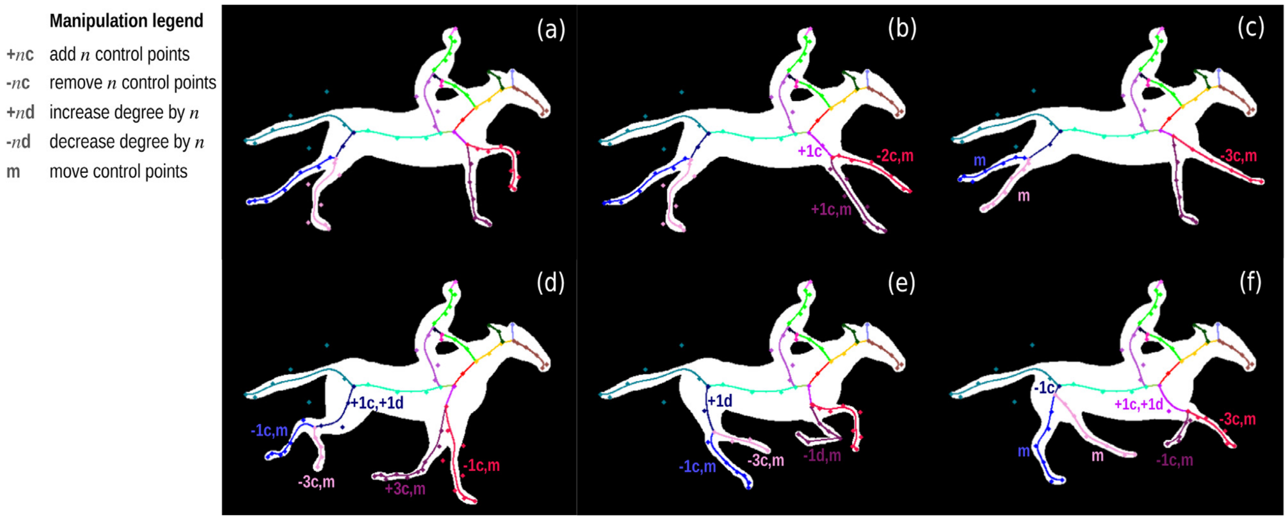

Fig. 13. Deformations of a running horse shape by manipulating the control points and the degree of B-splines. Control points and manipulations are colored just as the B-spline they affect.

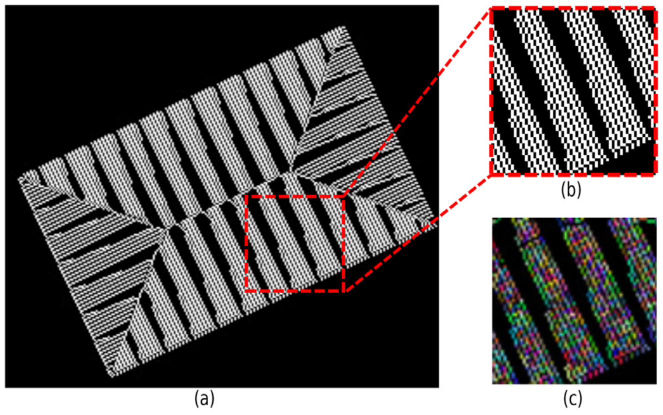

Fig. 14. The full medial axis of a rectangular shape (a) with zoomed-in detail (b) and color-coded segmented branches (c).

6. Discussion

We next discuss several aspects of our SMAT representation.

Ease of use: SMAT has two parameters that affect the trade-off between compression ratio and reconstruction accuracy: σ_0 and γ_0. Intuitively, σ_0 controls how many small-scale details (bumps) on the shape boundary are removed (MAT simplification); while γ_0 controls how much the MA branches are 'smoothed out' into splines (MAT approximation).

Speed: Since our MAT is implemented on the GPU (Section 2.1), the SMAT pipeline is very fast. For a shape of 512^2 pixels, the entire pipeline takes only a few tens of milliseconds on a Linux PC with an Nvidia RTX 2060 GPU. Following [41], the complexity of the SMAT computation is linear in the number of pixels in the input image.

Comparison: It is useful to summarize the differences between SMAT and Zhu et al. [16]. A key point is that Zhu et al. simplify MATs based on the *reconstruction error*, which naturally yields small errors. SMAT simplifies MATs based on how *salient* their points are for shape perception. Hence, our reconstructions can have a possibly larger Hausdorff distance H to the initial shape, as we are not explicitly optimizing for H. Both approaches are valid but for different goals: If one wants simplified MATs that reconstruct a shape as close as possible with respect to H, and is fine with computation times of minutes, Zhu et al. is to be used. If one accepts small reconstruction errors in non-salient shape parts, desires a multiscale MAT for simplification with different thresholds, and needs a real-time response, then our method is to be used.

Limitations: SMAT cannot (yet) be lossless, *i.e.*, yield an exact, zero Hausdorff-distance-to-original, reconstruction. To fully reconstruct the input shape, the *full*, unsimplified, MAT should be used. Fig. 14(a) shows the full medial axis of a simple rectangular shape. As Fig. 14(b) shows, the full-MA branches are very close. Algorithm 1 will segment this MA into tens of thousands of very short branches (shown in Fig. 14(c) with one color per such branch). The spline fitting (Sections 3.2.1 and 3.2.2) will approximate-and-merge such branches, resulting in a SMAT that cannot perfectly reconstruct the input shape. However, SMAT can achieve 0.3% approximation error and an MS-SSIM score of up to 0.99 (Section 4). We argue this is sufficient for most applications such as shape matching, retrieval, and deformation, which do not require perfectly lossless encoding.

7. Conclusion

We have presented SMAT, a method for encoding the medial axis transform (MAT) of raster (binary image) shapes with B-splines. For this, we simplify raster MATs using a salience metric, segment them into branches and branch-segments, and optimize the fit of a set of B-splines over the resulting segments to minimize the number of required control points while maximizing the spline-to-MAT fit. We evaluated SMAT on a collection of raster shapes of different types and complexities, using several quality metrics for image and shape comparison. Our results show that SMAT can achieve visually indistinguishable results to the original images, while reducing the space needed to store the MAT by one to two orders of magnitude. SMAT has only two simple-to-set parameters: the degree of MAT simplification and desired MAT approximation error. We showed how SMAT enables generating super-resolution images, can capture multiscale MATs, and allows shape manipulation. SMAT is implemented on the GPU making its application real-time for images up to 1000^2 pixels.

We next aim to extend SMAT beyond binary shapes, to encode grayscale and color images and potentially 2D and 3D scalar fields for scientific visualization applications. Separately, we aim to explore the potential of SMAT for image vectorization applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Jieying Wang: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Jiří Kosinka:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision. **Alexandru Telea:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision.

Acknowledgments

The first author acknowledges the China Scholarship Council (Grant number: 201806320354) for financial support.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cag.2021.05.012.

References

[1] Brady N. MPEG-4 standardized methods for the compression of arbitrarily shaped video objects. IEEE Trans Circuits Syst Video Technol 2000;9:1170–89.
[2] Freeman H. On the encoding of arbitrary geometric configurations. IRE Trans Electron Comput 1961;EC-10(2):260–8.
[3] Kim JI, Bovik AC, Evans BL. Generalized predictive binary shape coding using polygon approximation. Signal Process Image Commun 2000;15(7):643–63.
[4] Sánchez-Cruz H, Bribiesca E, Rodríguez-Dagnino RM. Efficiency of chain codes to represent binary objects. Pattern Recognit 2007;40(6):1660–74.
[5] Žalik B, Mongus D, Lukač N, Žalik KR. Efficient chain code compression with interpolative coding. Inf Sci (Ny) 2018;439–440:39–49.
[6] Figueiredo M, Leitao J, Jain A. Adaptive B-splines and boundary estimation. In: Proceedings of IEEE computer society conference on computer vision and pattern recognition; 1997. p. 724–30.
[7] Gerken P. Object-based analysis-synthesis coding of image sequences at very low bit rates. IEEE Trans Circuits Syst Video Technol 1994;4(3):228–35.
[8] Xiao Y, Jia Zou J, Yan H. An adaptive split-and-merge method for binary image contour data compression. Pattern Recognit Lett 2001;22(3):299–307.
[9] Aguilera-Aguilera E, Carmona-Poyato A, Madrid-Cuevas F, Marín-Jiménez M. Fast computation of optimal polygonal approximations of digital planar closed curves. Graph Models 2016;84:15–27.
[10] Aguilera-Aguilera E, Carmona-Poyato A, Madrid-Cuevas F, Medina-Carnicer R. The computation of polygonal approximations for 2d contours based on a concavity tree. J Vis Commun Image Represent 2014;25(8):1905–17.
[11] Yamaguchi N, Ida T, Watanabe T. A binary shape coding method using modified MMR. In: Proceedings of international conference on image processing, 1; 1997. p. 504–7.

[12] Brady N, Bossen F, Murphy N. Context-based arithmetic encoding of 2D shape sequences. In: Proceedings of international conference on image processing, 1; 1997. p. 29–32.
[13] Brady N. MPEG-4 standardized methods for the compression of arbitrarily shaped video objects. IEEE Trans Circuits Syst Video Technol 1999;9(8):1170–89.
[14] Kresch R, Malah D. Skeleton-based morphological coding of binary images. IEEE Trans Image Process 1998;7(10):1387–99.
[15] Wang H, Schuster GM, Katsaggelos AK, Pappas TN. An efficient rate-distortion optimal shape coding approach utilizing a skeleton-based decomposition. IEEE Trans Image Process 2003a;12(10):1181–93.
[16] Zhu Y, Sun F, Choi Y-K, Jüttler B, Wang W. Computing a compact spline representation of the medial axis transform of a 2D shape. Graph Models 2014;76(5):252–62.
[17] Yushkevich P, Thomas Fletcher P, Joshi S, Thall A, Pizer SM. Continuous medial representations for geometric object modeling in 2D and 3D. Image Vis Comput 2003;21(1):17–27.
[18] Blum H. A transformation for extracting new descriptors of shape. In: Models for the perception of speech and visual form. Cambridge: MIT Press; 1967. p. 362–80.
[19] Blum H, Nagel RN. Shape description using weighted symmetric axis features. Pattern Recognit 1978;10(3):167–80.
[20] Meijster A, Roerdink J, Hesselink W. A general algorithm for computing distance transforms in linear time. In: Mathematical morphology and its applications to image and signal processing. Springer; 2002. p. 331–40.
[21] Hesselink WH, Roerdink JBTM. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. IEEE TPAMI 2008;30(12):2204–17.
[22] Pizer S, Siddiqi K, Székely G, Damon J, Zucker S. Multiscale medial loci and their properties. IJCV 2003a;55:155–79.
[23] Siddiqi K, Pizer S. Medial representations: mathematics, algorithms and applications. 1st ed. Springer Publishing Company, Incorporated; 2008. ISBN 1402086571.
[24] Saha PK, Borgefors G, Sanniti di Baja G. A survey on skeletonization algorithms and their applications. Pattern Recognit Lett 2016;76:3–12.
[25] Lam L, Lee S , Suen CY. Thinning methodologies-a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 1992;14(9):869–85.
[26] Ogniewicz R, Kübler O. Hierarchic Voronoi skeletons. Pattern Recognit 1995;28(3):343–59.
[27] Attali D, Montanvert A. Computing and simplifying 2D and 3D continuous skeletons. Comput Vision Image Understanding 1997;67(3):261–73.
[28] Amenta N, Bern M. Surface reconstruction by Voronoi filtering. Discret. Comput Geomet 1999;22(4):481–504.
[29] Kimmel R, Shaked D, Kiryati N, Bruckstein AM. Skeletonization via distance maps and level sets. Comput Vision Image Understanding 1995;62(3):382–91.
[30] Sethian JA. A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. 1996;93(4):1591–5.
[31] Telea A, Wijkvan J. An augmented fast marching method for computing skeletons and centerlines. In: Proceedings of the symposium on data visualization. Eurographics; 2002. p. 251–9.
[32] Falcão AX, Stolfi J, de Alencar Lotufo R. The image foresting transform: theory, algorithms, and applications. IEEE Trans Pattern Anal Mach Intell 2004;26(1):19–29.
[33] Zwan M, Meiburg Y, Telea A. A dense medial descriptor for image analysis. In: VISAPP 2013 - proceedings of the international conference on computer vision theory and applications, 1; 2013. p. 285–93.
[34] Cao T-T, Tang K, Mohamed A, Tan T-S. Parallel banding algorithm to compute exact distance transform with the GPU. In: Proceedings of the 2010 ACM SIGGRAPH symposium on interactive 3D graphics and games. New York, NY, USA: Association for Computing Machinery; 2010. p. 83–90.
[35] Telea A. Feature preserving smoothing of shapes using saliency skeletons. In: Proc. VMLS. Springer; 2012. p. 153–70.
[36] Attali D, Montanvert A. Modeling noise for a better simplification of skeletons. In: Proc. IEEE ICIP, vol. 3; 1996. p. 13–16.
[37] Foskey M, Lin M, Manocha D. Efficient computation of a simplified medial axis. J Comput Inf Sci Eng 2003;3(4):274–84.
[38] Dey T, Zhao W. Approximate medial axis as a Voronoi subcomplex. Comput Aided Des 2004;36(2):195–202.
[39] Siddiqi K, Bouix S, Tannenbaum AR, Zucker SW. Hamilton-Jacobi skeletons. IJCV 2002;48(3):215–31.
[40] Tagliasacchi A, Delame T, Spagnuolo M, Amenta N, Telea A. 3D Skeletons: a state-of-the-art report. Comput Graph Forum 2016;35(2):573–97.
[41] Wang J, Terpstra M, Kosinka J, Telea A. Quantitative evaluation of dense skeletons for image compression. Information 2020;11(5):274.
[42] Wang J, Joao L, Falcão A, Kosinka J, Telea A. Focus-and-context skeleton-based image simplification using saliency maps. In: Proceedings of the 16th International joint conference on computer vision, imaging and computer graphics theory and applications - Volume 4: VISAPP. SciTePress; 2021. p. 45–55.
[43] Attali D, Boissonnat J-D, Edelsbrunner H. Stability and computation of medial axes-a state-of-the-art report. In: Mathematical foundations of scientific visualization, computer graphics, and massive data exploration. Springer; 2009. p. 109–25.
[44] Rote G. Computing the minimum Hausdorff distance between two point sets on a line under translation. Inf Process Lett 1991;38(3):123–7.
[45] Zhang D, Lu G. Review of shape representation and description techniques. Pattern Recognit 2004;37:1–19.
[46] Jaccard P. The distribution of flora in the alpine zone. New Phytol 1912;11(2):37–50.
[47] Wang Z, Simoncelli E, Bovik A. Multiscale structural similarity for image quality assessment. In: Proc. asilomar conf. on signals, systems computers; 2003b. p. 1398–402.
[48] Wang Z, Bovik A, Sheikh H, Simoncelli E. Image quality assessment: from error visibility to structural similarity. IEEE TIP 2004;13:600–12.
[49] Marschner S, Shirley P. Fundamentals of computer graphics. 4th ed. USA: A. K. Peters, Ltd.; 2016.
[50] Piegl L, Tiller W. The NURBS book (2nd ed.). Berlin, Heidelberg: Springer-Verlag; 1997. ISBN 3540615458.
[51] Eberly D. Least-squares fitting of data with b-spline curves. 2014. Geometric Tools. www.geometrictools.com/Documentation/BSplineCurveLeastSquaresFit.pdf.
[52] Kosinka J, Jüttler B. G1 Hermite interpolation by minkowski pythagorean hodograph cubics. Comput Aided Geom Des 2006;23(5):401–18.
[53] Cohen E, Lyche T, Riesenfeld R. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. Comput Graph Image Process 1980;14(2):87–111.
[54] Telea A. CUDASkel: real-time computation of exact euclidean multiscale skeletons on CUDA. 2019. webspace.science.uu.nl/~telea001/Shapes/CUDASkel.
[55] Wang, Jieying and Kosinka, Jiři and Telea, Alexandru. SMAT source code and datasets. 2021. https://github.com/WangJieying/SMAT_code.
[56] Ralph R. MPEG-7 dataset. 2019. http://www.dabi.temple.edu/~shape/MPEG7/dataset.html.
[57] Strzodka R, Telea A. Generalized distance transforms and skeletons in graphics hardware. In: VisSym 2004, Symposium on Visualization. Eurographics Association; 2004. p. 221–30.
[58] Sýkora D, Dingliana J, Collins S. As-rigid-as-possible image registration for hand-drawn cartoon animations. In: Proceedings of the 7th international symposium on non-photorealistic animation and rendering. New York, NY, USA: Association for Computing Machinery; 2009. p. 25–33.
[59] Weng Y, Xu W, Wu Y, Zhou K, Guo B. 2D shape deformation using nonlinear least squares optimization. Vis Comput 2006;22:653–60.
[60] Mota T, Esperana C, Oliveira A. 2D shape deformation based on positional constraints and layer manipulation. In: 2011 Brazilian symposium on games and digital entertainment; 2011. p. 1–10.
[61] Pizer S, Fletcher P, Joshi S, Thall A, Chen J, Fridman Y, Fritsch D, Gash A, Glotzer J, Jiroutek M, Lu C, Muller K, Tractor G, Yushkevich P, Chaney E. Deformable M-Reps for 3D medical image segmentation. IJCV 2003b;55(2–3):85–106.
[62] Angelidis A, Cani MP. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In: Proc. ACM SMA; 2002. p. 45–52.
[63] Yoshizawa S, Belyaev A, Seidel HP. Skeleton-based variational mesh deformations. CGF 2007;26(3):255–64.