TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Topologically safe curved schematisation

Citation for published version (APA):
Goethem, van, A. I., Meulemans, W., Reimer, A., Haverkort, H. J., & Speckmann, B. (2013). Topologically safe curved schematisation. *The Cartographic Journal*, *50*(3), 276-285. https://doi.org/10.1179/1743277413Y.0000000066

DOI:
10.1179/1743277413Y.0000000066

Document status and date:
Published: 01/01/2013

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 09. Oct. 2026

Topologically Safe Curved Schematization*

Arthur van Goethem† Wouter Meulemans† Andreas Reimer‡
Herman Haverkort† Bettina Speckmann†

Abstract

Traditionally schematized maps make extensive use of curves. However, automated methods for schematization are mostly restricted to straight lines. We present a generic framework for topology-preserving curved schematization that allows a choice of quality measures and curve types. The framework fits a curve to every part of the input. It uses Voronoi diagrams to ensure that curves fitted to disjoint parts do not intersect. The framework then employs a dynamic program to find an optimal schematization using the fitted curves. Our fully-automated approach does not need critical points or salient features. We illustrate our framework with Bézier curves and circular arcs.

1 Introduction

Schematization is a design-rule driven effort to minimize non-functional detail according to a target complexity. It is a special case of cartographic generalization, one where the radical law (Töpfer & Pillewizer 1966) does not directly apply. So far, most research on automated schematization has concentrated on straight line segments with restrictions on the admissible directions (e.g. Buchin et al. 2011, Cicerone & Cermignani 2012). Many manually produced schematized maps, however, make copious use of curves, see Fig. 1 and the work of Reimer (2010). Curves have greater expressive power than line segments: several line segments can often be replaced by a single, low-degree curve.

Software for automated map production and geodata handling only recently started to utilize parametric curves. If curve fitting is used as a schematization operation such as caricature or smoothing (Regnauld & McMaster 2007), we speak of curve approximation. Curve approximation is known to depend strongly on the use case and, hence, no general solution exists (Piegl & Tiller 1997). While spatial DBMS, GIS, and CAD/CAM infrastructures have begun to merge (for

*A. van Goethem, B. Speckmann, and W. Meulemans are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.102 (AvG) and no. 639.022.707 (BS and WM).
An abstract of this paper appeared in the collection of abstracts of the 29th European Workshop on Computational Geometry (EuroCG), pp. 87-90, 2013.

†Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands, a.i.v.goethem@tue.nl, cs.herman@haverkort.net, w.meulemans@tue.nl, and speckman@win.tue.nl
‡Institute of Geography, University of Heidelberg, Germany, andreas.reimer@geog.uni-heidelberg.de

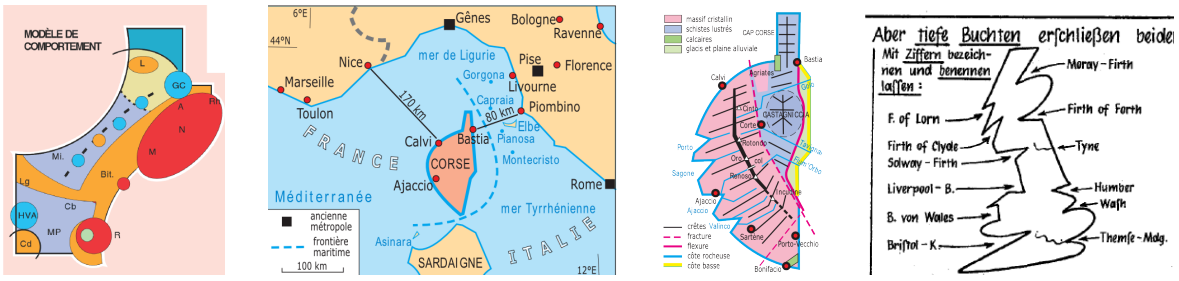

Figure 1: Manually drawn schematizations of Languedoc-Roussillon (Brunet 1991), two times Corsica (Brunet 2004), and Britain (Gürtler 1929).

example, native Bézier curves in ArcGIS 10), concrete applications in automated cartography using curved generalizations or schematizations are still missing (Bucher & Schlömer 2006, Roberts 2012).

When using curves in cartography we need topologically sound results. Generalization and other geo-processing operations are often implemented in recursive processes that repeatedly make expensive checks for topology violations. The removal of detected violations is a challenge and often triggers more checks. To the best of our knowledge, the framework described in this paper is the first that ensures topology preservation for curved schematization operations while avoiding repeated checks for topology violations.

Contribution. We describe a generic framework for computing curved schematizations of simple polygons and planar subdivisions ('multiple polygons'). The framework requires two components to be specified. The first is a method to fit a curve to a chain, which consist of one or more consecutive edges of the input; the second is a distance measure expressing how well the curve fits. Our framework segments the input at a subset of its vertices and fits curves to the resulting polygonal chains. To ensure that the result of this operation is topologically correct, we use the Voronoi cell of each chain and constrain the curves to remain within these cells.

The quality of the schematization depends on the chosen segmentation. We formulate the problem of finding the best segmentation as an optimization problem. Hence we do not need critical points or salient features to be specified by the user or detected beforehand.

The framework is designed specifically for territorial outlines. Nevertheless, it can be applied to any planar object defined by polygonal lines, such as thematic information, rivers and lakes, or road networks.

Organization. Below we first review related work. In Section 2 we introduce our framework for simple polygons. Section 2.1 discusses how to ensure that curves are fitted in a topologically safe way and then Section 2.2 shows how to find the optimal segmentation using dynamic programming. To illustrate our framework we give example implementations with Bézier curves and circular arcs (Section 3 and 4, respectively). Section 5 describes the extension of our framework to subdivisions. Finally in Section 6 we discuss our results and possible future work.

Related work. Automated simplification and smoothing with *straight lines* have received significant attention over the years. Mustafa et al. (2001) use Voronoi cells for topologically safe simplification and heuristics to speed up the process. Our framework yields similar results when fitting straight lines as degenerate curves and using the directed Hausdorff distance as a distance measure. Van der Poorten and Jones (2002) use constrained Delaunay triangulations to find and simplify features in a topologically safe way.

Existing work on automated schematization of *shapes* is sparse. So far it has concentrated on straight-line drawings. For example Buchin et al. (2011) describe an area-preserving schematization method that yields straight-line results in which each line segment adheres to a given set of orientations. Cicerone and Cermignani (2012) give an algorithm to generate rectilinear or octagonal schematizations. Reimer and Meulemans (2011) conjecture that parallelism drives the schematization. They give a heuristic method to optimize a parallelism measure. De Chiara et al. (2011) use an even less restricted model of straight-line schematization, equating it to simplification. The related concept of curved abstraction has been researched in the field of non-photorealistic rendering. In the paper by Mi et al. (2009) a shape is decomposed in basic parts and reconstructed up to a given detail. Their method in fact uses curves. In contrast to our work, their work is not concerned with topological correctness.

A variety of methods fit a (cubic) Bézier curve to a polygonal line (Masood & Ejaz 2010, Schneider 1990, Shao & Zhou 1996). However, these methods often do not avoid intersections nor is it obvious how to adapt the fitting process to avoid intersections when fitting multiple curves. Schneider (1990) applies a heuristic similar to Douglas-Peucker to fit multiple curves, in essence using critical points. Shao and Zhou (1996) also use critical point detection before fitting Bézier curves. B-splines have been applied to approximate polygons (Guilbert & Lin 2007, Saux & Daniel 1999). Intersections

of different splines, however, still need to be checked and resolved separately.

Drysdale et al. (2008) present an algorithm to compute an approximation with a minimal number of circular arcs for a given tolerance region and a set of 'gates'. However, requirements on these gates prevent a significant complexity reduction. Heimlich and Held (2008) describe how to generate smooth approximations with circular arcs using tolerance bands. Their framework allows other curves, but cannot guarantee optimal results. Also, smooth approximations are not always suitable for schematization. Neither of these methods can incorporate area-preservation constraints.

2 The framework

The basic version of our framework generates a curved schematization of an input simple polygon. We refer to one or more consecutive edges of this polygon as a *chain*. The polygon is segmented at its vertices resulting in a number of chains. A curve is fit to each of these chains. To ensure that the result is topologically correct, we use the Voronoi cells (De Berg et al. 2008) of each chain. That is, the curve fitted to a chain is constrained to remain within its Voronoi cell. As a result, every curve lies closer to its chain than to any other part of the polygon. A dynamic program ensures that the optimal segmentation is selected.

To use the framework two components need to be specified. The first component is a method to fit a simple curve to a chain, that is, to a polygonal line. There are two requirements on a simple curve fit. First, the curve must start and end at the endpoints of the corresponding chain. Second, the curve may not have self-intersections. Any type of curve can be used, for example, circular arcs, Bézier curves, or even simply straight lines. The second component is a distance measure which expresses how well the curve fits. The distance measure can be any function that assigns a non-negative value to the combination of a curve and a chain. A low value should indicate a high quality of fit.

2.1 Topologically safe fitting

We first compute the edge-based Voronoi diagram of the input polygon. This diagram associates with every edge of the polygon a cell of the diagram. A variety of algorithms exist to compute the edge-based Voronoi diagram, for example, the algorithm of Lee and Drysdale (1981). Every Voronoi cell is a simple polygon without holes (but potentially unbounded and with parabolic parts). The Voronoi cell of a chain is the union of the Voronoi cells of its edges (see Fig. 2). To guarantee topological correctness, we ensure that the union of cells remains a simple polygon without holes. Consider the chain of white vertices in Fig. 3 (a). The union of the corresponding Voronoi cells contains a hole. Consequently, a non-topologically safe curve may be fitted. By enforcing a boundary in such a scenario, a non-topologically safe curve is not considered to be inside the union of Voronoi cells. Topologically safe curves are not affected by this change (see Fig. 3 (b)).

To be able to quickly change the parameters of the schematization, we pre-compute a two-dimensional table T. Let $P[i, j]$ denote the chain of the input polygon from vertex i to vertex j.

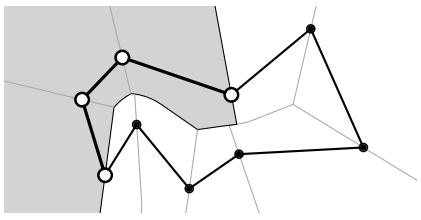

Figure 2: A simple polygon and its Voronoi cells. The union cell of a chain of three edges is marked.

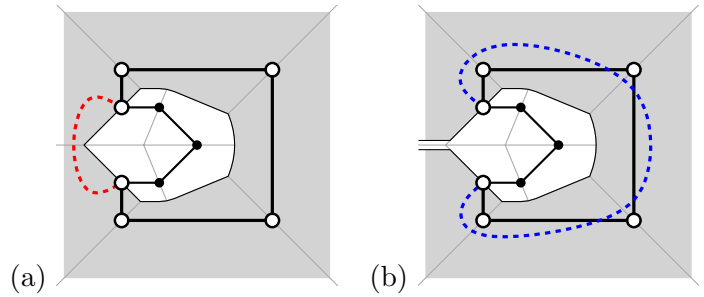

Figure 3: (a) Dashed curve lies in the cell but violates topology. (b) Union enforces correct topology.

For $i = j$ this is the entire polygon. Each entry $T[i, j]$ in the table contains the curve fitted to $P[i, j]$ and a value that indicates the quality of the fit. For each entry of T we compute the associated cell (possibly a union of Voronoi cells) and fit a curve to the chain. If the curve lies within the cell, we compute the quality of the fit and store it. If not, we disallow the use of this curve to ensure topological correctness and set $T[i, j] = \infty$. Note that the curve is not required to pass through the Voronoi cell of each separate edge of the chain.

Let $F(v)$ represent the time required to fit a curve to a chain with v vertices, check whether the curve lies within the chain's cell, and compute the quality of the fit. Let n denote the number of vertices of the polygon. Since the table has n^2 entries, it takes $O(n^2 F(n))$ time to compute the table.

2.2 Optimal segmentation

To obtain a schematization from the table T, we select the curves representing a collection of consecutive chains that together constitute the complete polygon. We distinguish two variants, as described below. We first explain, for each of these variants, how to compute the value of the optimal solution, assuming that we start at vertex 1. In the presentation of both variants we assume that $T[i, j]$ denotes only the quality of fit of the curve fitted to $P[i, j]$. For simplicity of explanation we define $T[j, n + 1] = T[j, 1]$.

Min-# problem. For the min-# problem, we wish to minimize the number of curves that are used, if the distance of each curve to its segment is restricted to be at most ϵ. The minimum number of curves needed can be determined by computing the values of the following expression for $i = 1$ up to $i = n + 1$:

$$OPT[i] = \begin{cases} 0, & \text{if } i = 1 \\ \min_{1 \le j < i \text{ and } T[j,i] \le \epsilon} OPT[j] + 1, & \text{if } i > 1 \end{cases}$$

The computation takes $O(n^2)$ time and the end result is $OPT[n + 1]$. A solution is guaranteed to exist if the curve representing a single line segment is the line segment itself and has distance zero.

Min-ϵ problem. For the min-ϵ problem, we wish to minimize the maximum distance of the selected curves, while using at most K curves. The minimum achievable distance can be determined by computing the values of the following expression for $i = 1$ up to $i = n + 1$ and for $k = 1$ up to $k = K$:

$$OPT[i, k] = \begin{cases} 0, & \text{if } i = 1 \\ T[1, i], & \text{if } k = 1 \text{ and } i > 1 \\ \min \left\{ \begin{array}{c} OPT[i, k-1], \\ \min_{1 \le j < i} \max \{T[j, i], OPT[j, k-1]\} \end{array} \right\}, & \text{if } k > 1 \text{ and } i > 1 \end{cases}$$

The computation takes $O(n^2K)$ time and the end result is $OPT[n+1, K]$. A solution is guaranteed to exist if the curve fitting method can fit a curve to a chain that starts and ends at the same point (i.e. the entire polygon with a given start vertex). We note that the first case of the expression is only necessary because of our assumption that $T[i, i]$ represents the entire polygon.

Obtaining a schematization. To compute the best solution, we repeat the above computations for each vertex as starting vertex. We obtain the actual schematization by keeping track of the choices made during the computation of OPT. The initialized table $OPT[i, k]$ for the min-ϵ problem is independent of the scale or complexity parameter. Hence, one table can be used to create multiple schematizations with different parameter values. As a byproduct of querying for K curves, we obtain all results using less than K curves. So if we run the query for $K = n$ once, we obtain all possible schematizations and we can store these instead of the table. A query for different values then becomes a simple lookup. We can also handle queries for the min-# problem by computing the maximum distance used in each of the solutions and performing a binary search.

3 Cubic Bézier curves

In this section we show how to use our framework with Bézier curves. We describe a method to fit a Bézier curve to a chain and specify a distance measure.

Distance measure. Assume that we are given a Bézier curve C and a chain S with m vertices. In contrast to fitting a curve to a set of data points, we want to fit a curve to a chain. Therefore, we base our distance measure on a dense sampling of S. These samples are regularly spaced in the parameter space of C. Each of these samples can be matched with a corresponding point along S as parameterized by length.

We desire long curves that approximate the data well, since such curves can frequently be observed in manually drawn curved schematizations. The distance measure of C is, therefore, a weighted average of the squared distance between corresponding points, divided by the length of S. Formally, our distance measure between curve C and chain S is:

$$\frac{\sum_{i=0}^{2m} \|C(i/2m) - S(i/2m)\|^2 * (i-m)^2}{length(S) * \sum_{i=0}^{2m} (i-m)^2} \text{ , where}$$

$length(S)$: Euclidean length of S;

$C(t), S(t)$: position on C or S, as parameterized by t;

$\|u - v\|$: Euclidean distance between points u and v.

When curves do not match the local contour at their endpoints, the unweighted variant may create visually salient points not present in the input data (see Fig. 4 (a)). To avoid such visual artifacts we use the weighted average as given above that assigns a high weight to samples near the endpoints and a low weight to the midsection. As a result, features along the midsections of curves may disappear (see Fig. 4 (b)). We deem the disappearance of features less disturbing than the appearance of features that do not exist.

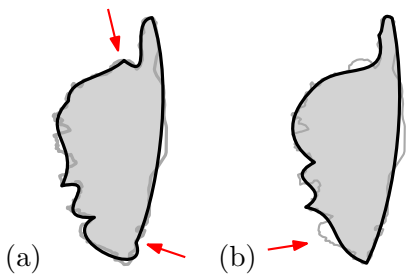

Figure 4: (a) Regular squared error. (b) Weighted squared error.

To exclude self-intersecting curves we define the distance measure of a self-intersecting curve to be infinity. Self-intersection of a cubic Bézier curve is tested by examining the control points (Stone & DeRose 1989).

Fitting a cubic Bézier curve. For fixed tangents at the endpoints, Schneider (1990) shows how to algebraically compute the Bézier curve optimizing the least-squares measure in linear time. A similar method can be used to optimize our weighted measure.

We direct the initial tangents towards the furthest vertex on either side of the straight line connecting the endpoints (Masood & Ejaz 2010). Let points P_0, P_1, P_2, and P_3 be the control points of the Bézier curve. The straight line connecting the endpoints of the chain S (which define P_0 and P_3) splits the plane in two half-planes (see Fig. 5). Define p and s as the vertices of S furthest from the line l containing this chord in either half-plane. If one of the half-planes does not contain any vertices of S, then we let both p and s be the point furthest from l in the other half-plane. We assume p occurs at or before s when we traverse S from P_0 to P_3. We choose the tangent at P_0 to be directed to p and the tangent at P_3 to be directed to s. The rationale for this choice is that these extreme points are likely to be salient points, and therefore the curve should bend in their direction.

To compute the squared distance each vertex on the chain S needs to be matched to a point on the curve. To this end, we parameterize each vertex with a value t between zero and one. This value directly matches the vertex to a unique point on the Bézier curve, that is, to $(1-t)^3 \cdot P_0 + 3 \cdot (1-t)^2 \cdot t \cdot P_1 + 3 \cdot (1-t) \cdot t^2 \cdot P_2 + t^3 \cdot P_3$. Given this matching we can fit the Bézier curve optimally. When fitting the Bézier curve we compute the optimal matching between the vertices and the curve. Please note that this matching between the vertices and the curve is different from the matching described before used to compute the distance measure. To prevent overfitting on the vertices the matching used for the distance measure is simply a uniform matching of the chain S with the curve.

Algebraically computing an optimal matching between the vertices and the curve is impossible. Instead, we use the chord-length parameterization (Ahlberg et al. 1967) as an initial approximation. During the following steps, we optimize the parameterization with numerical methods.

We apply the algorithm by Schneider (1990) to algebraically compute the optimal position for control points P_1 and P_2 on their respective tangent lines. This process iteratively improves the parameterization of vertices using Newton-Raphson and computes optimal positions given the current parameterization. After sufficient iterations we obtain a good approximation of the optimal position along the tangent lines.

As a subsequent step we fix the distance between control points P_0 and P_1, and between P_2 and P_3. We now optimize the tangents for either side algebraically. Once more we use Newton-Raphson approximation to obtain a better parameterization. If desired, the process of subsequently

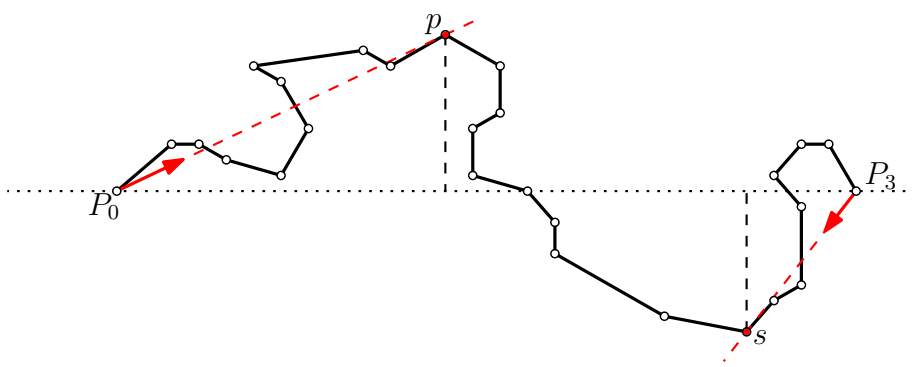

Figure 5: We guess the initial tangents based on the extremal points compared to the line through the first and last vertex of the chain.

optimizing the distance and tangents can be repeated. In our experiments, a few iterations were sufficient to obtain a stable solution.

Topological validity is tested by checking for intersections between a polygonal approximation of the Voronoi cell and the convex hull of the control points. This intersection check is conservative and will always detect an intersection if there is one. However, we may obtain a false positive and detect an intersection that is not there. To obtain a less conservative intersection test the curve can be subdivided a constant number of times using De Casteljau's algorithm (De Casteljau 1959) to obtain a tighter fit around the curve.

Algorithmic complexity. Fitting a Bézier curve takes $O(n)$ time for a single iteration. We repeat the process until the solution is stable, which takes 3 to 6 iterations in our experiments. Thus, assuming a constant upper bound on the number of iterations, the time required to fit a single curve is $F(n) = O(n)$.

Results. Figures 6, 7, and 8 show results of using the methods above with our framework. Each figure depicts schematizations of the same shape with different complexities. Even a small number of curves result in recognizable and pleasing shapes. Note that the detail in the schematization is not necessarily homogeneously distributed along the representation (see, for example, Fig. 8). The algorithm selects detail depending on the local saliency of a feature, which is the desired behavior.

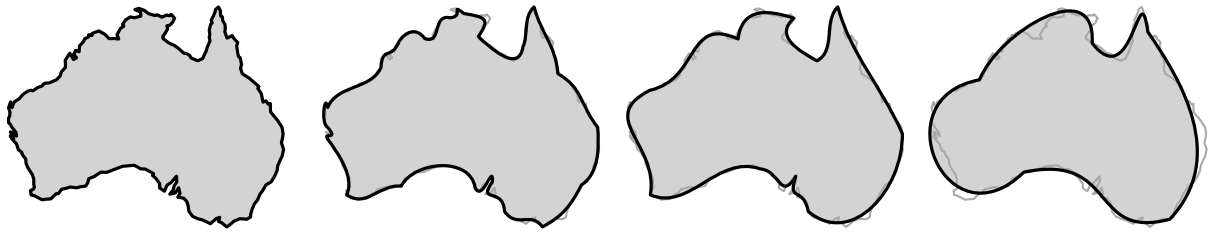

Figure 6: Australia with 20, 10, and 5 curves.

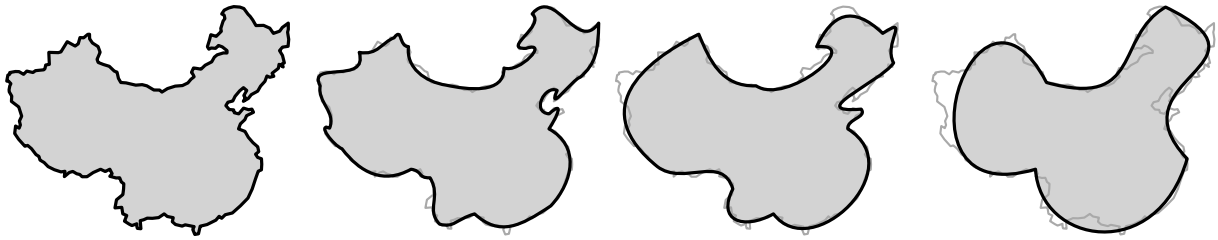

Figure 7: China with 20, 10, and 4 curves.

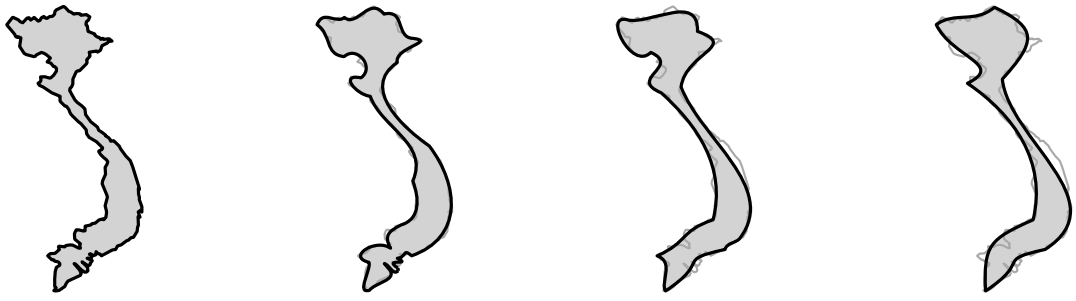

Figure 8: Vietnam with 20, 10, and 7 curves.

4 Circular arcs

Inspired by recent work on area-preserving schematization (Buchin et al. 2011) and circular-arc graph drawing (Duncan et al. 2012), we use our framework for area-preserving circular-arc schematizations. Though unusual in cartography, circular arcs are used in mnemonic sketches for school use, known as merkbilder and croquis, as well as in choromatic diagrams.

Fitting arcs. Given a chain C that starts at u_1 and ends at u_k, the circular arc that locally preserves the area is unique. To this end, we consider the *signed area* of a chain. Let polygon P' correspond to the full polygon P in which C has been replaced by a straight line. The signed area of C is then the area of P minus the area of P' (see Fig. 9). Let Δ denote the signed area of C. To compute Δ it is not necessary to explicitly compute the polygon areas. Instead, the following formula suffices:

$$\Delta = \frac{1}{2}(u_k \times u_1 + \sum_{i=1}^{k-1} u_i \times u_{i+1}) \ .$$

In this formula, $\times$ indicates the 2-dimensional vector cross product, that is, $(x_1, y_1) \times (x_2, y_2) = x_1 \cdot y_2 - x_2 \cdot y_1$. We now need to find an arc from u_1 to u_k that has the same signed area Δ as the chain C it replaces. This is computed by solving:

$$\frac{8 \cdot \Delta}{d^2} = \frac{\alpha - \sin\alpha}{\sin^2 \frac{\alpha}{2}} \ ,$$

where d is the distance between u and v, and α is the central angle of the desired arc. The central angle α uniquely determines an arc and it is straightforward to compute the arc from α. However, computing α from d and Δ seems algebraically impossible. Fortunately, the righthand side of the equation is monotone and we can use numerical methods to compute α approximately. This monotonicity also implies that the desired arc is unique.

Scoring arcs. The distance measure we use with our framework is the continuous Fréchet distance, extended for curves (Rote 2007). It takes $O(m \log m)$ time to compute this distance between a single arc and a chain of length m. Hence, we conclude that $F(n) = O(n \log n)$.

Results. Figures 10, 11, and 12 show some results using area-preserving circular arcs. A very low number of arcs results in a very stylized shape. The results typically retain some features, but are hard to recognize without context such as a map title. A few more arcs give a stylized or playful appearance and make the shape more recognizable. Using only 20 arcs results in a very good approximation of the input while retaining the playful appearance.

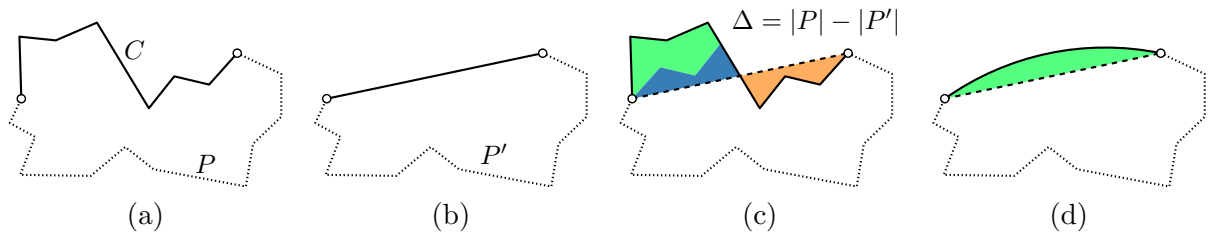

Figure 9: (a) A polygon P with chain C. (b) Polygon P' obtained from replacing C in P with a straight line. (c) Signed area Δ obtained from the difference in area. The negative orange area and positive blue area sum to zero. Thus, the green area equals the signed area. (d) The arc with the same signed area as chain C.

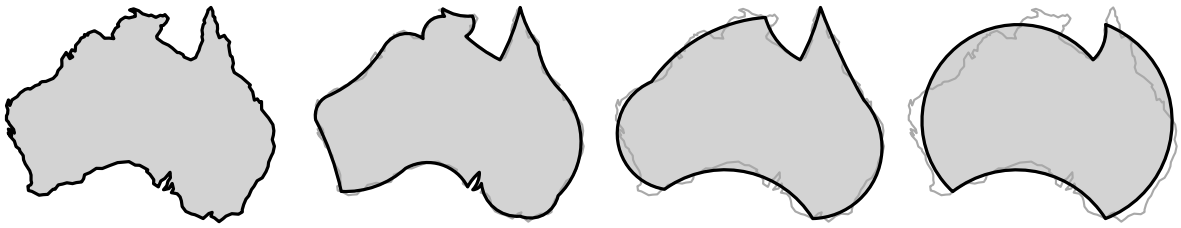

Figure 10: Australia with 20, 7, and 4 arcs.

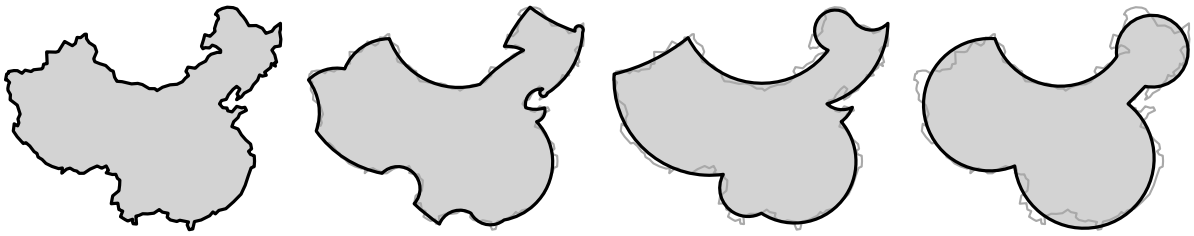

Figure 11: China with 20, 10, and 5 arcs.

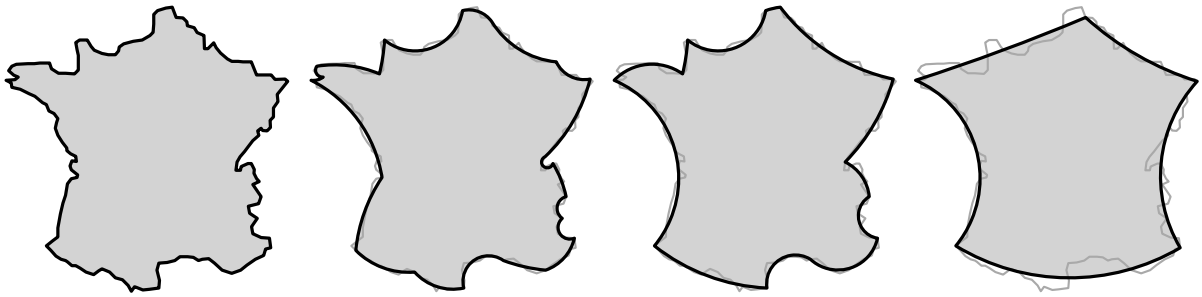

Figure 12: France with 20, 12, and 5 arcs.

5 Extension to subdivisions

The basic version of our framework takes as input a single simple polygon. A more general structure is a *subdivision*. Similar to a polygon, it is defined by vertices and straight-line edges. In a polygon, each vertex has exactly two incident edges. In contrast, a vertex in a subdivision may have multiple incident edges. The number of incident edges is called the *degree* of a vertex. We define a *boundary* as a maximum sequence of edges such that all its vertices have degree two, with the possible exception of its endpoints. There are two types of boundaries, circular and non-circular boundaries. The former corresponds to a polygon and has only vertices of degree 2. The latter corresponds to a chain and ends at vertices of degree 3 or higher. Considering a subdivision representing territorial outlines, a circular boundary would correspond to an island; a non-circular boundary would correspond to a shoreline or border between two territories.

It is straightforward to extend our framework to support subdivisions. The Voronoi diagram is computed for the entire subdivision. However, curves are fitted only to chains that are part of a single boundary. The min-# problem requires no change. It can be applied to every boundary in isolation to compute the global optimal. Note, however, that we need not test all starting vertices for a non-circular boundary: the starting vertex is required to be either of the endpoints.

The min-ϵ problem does require some changes. Boundaries are not independent subproblems as the number of curves allowed is a global requirement. The number of curves used for one boundary affects the number of curves that may be used by others. A second dynamic program can resolve this issue. Suppose that the subdivision consists of B boundaries, numbered 1 to B. Using the

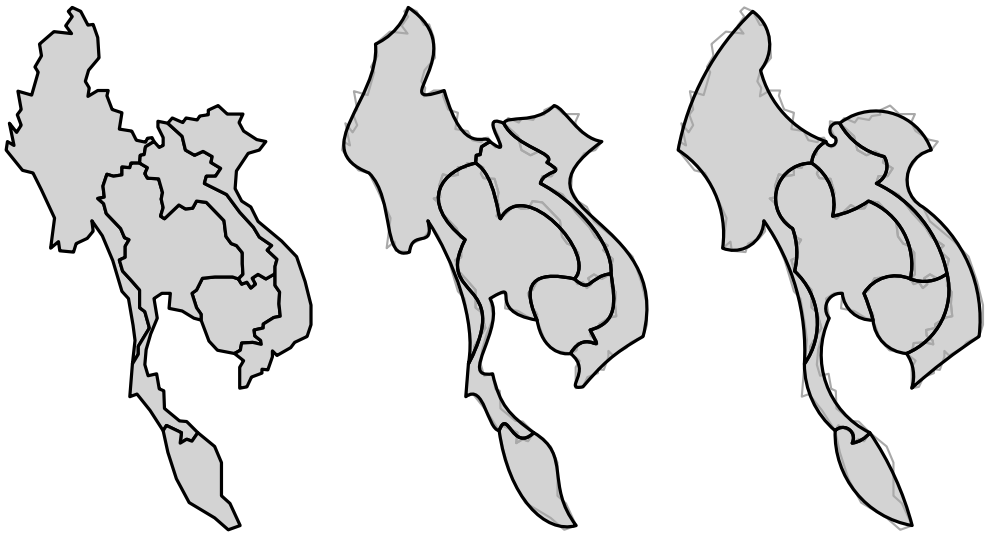

Figure 13: Part of southeast Asia: Cambodia, Laos, Myanmar, Thailand, Vietnam, and part of Malaysia. Result with 30 Bézier curves (middle) and with 30 circular arcs (right).

min-ϵ formulation of Section 2.2, we can compute the optimal solution for each boundary b given a number of curves k. Let $S[b,k]$ denote this solution. We can now compute the optimal solution for the subdivision using at most K curves by solving the following for $b = 1$ up to $b = B$ and $k = 1$ up to $k = K$:

$$OPT[b,k] = \begin{cases} S[b,k], & \text{if } b = 1 \\ \infty, & \text{if } k < b \\ \min_{1 \le j \le k-b+1} \max\{OPT[b-1, k-j], S[b,j]\}, & \text{if } k \ge b > 1 \end{cases}$$

Note that the second case in this equation is actually never needed, provided $K \ge B$. We list it for completeness and to indicate that it is never possible to obtain a schematization with less curves than boundaries. Some results for subdivisions are illustrated in Fig. 13.

6 Discussion and conclusion

We presented a framework for topologically safe schematization. Our framework is fully automated and does not require critical point detection. We illustrated the framework using cubic Bézier curves and circular arcs. Both cases indicate that high-quality results can be obtained using our framework and that the restriction to Voronoi cells (to preserve topology) is not too restrictive. Fig. 14 shows the potential of our schematizations for use in diagrams.

Schematizations can adhere to different design principles such as emphasis through size (e.g. cartograms), cartographic smoothing with parametric curves (e.g. chorematic diagrams such as Corsica in Fig. 1) or stylization via angular restriction (e.g. octagonal metro maps). In this work, we have shown the usefulness of the framework for application in smoothing schematizations using Bézier curves and for stylized schematization using circular arcs. Both variants convey different visual impressions and have different cartographic properties: with the same number of segments, the Bézier-curve schematization is closer to the original shape, whereas the circular-arc solutions more strongly convey rigorousity, abstraction, and artificiality. This is reinforced by the area-preserving, i.e. conformal nature of the circular-arc approach.

Drawbacks. The described framework has two main drawbacks. The first is that it does not allow vertices to be moved. That is, the vertices of the schematization are a subset of the vertices of the

input polygon. For schematization, it is often undesirable to have vertex-restricted results. Being limited to the input vertices lacks the expressive power of introducing new vertices. In particular, vertex restriction poses problems for very low complexities and subdivisions.

The second drawback is that our framework might not find low-complexity results for very intricate shapes. This issue may arise if two or more parts are geometrically close together, but far apart along the polygon boundary. The Voronoi cells can be overly restrictive, thus not allowing a low-complexity schematization. But if we relax the Voronoi-cell constraint, we can no longer guarantee that results are topologically correct. However, this problem does not seem to arise frequently for our intended shapes, that is, for territorial outlines.

Future work. Our fully-automated approach comes at a price: the pre-processing stage of our framework is very time-consuming. Using critical points would speed up our algorithm significantly. However, it is unclear how to choose critical points whose inclusion leads to the same high-quality results as our current approach. Hence other methods to speed up computation need to be explored.

It would also be interesting to further explore continuity between curves. The current framework ensures only that the curves touch at their endpoints. However, in many cases, small angles may be undesirable and curves should smoothly continue from one to the other. In part, this effect was achieved for cubic Bézier curves by weighing the vertices, though small discontinuities continue to exist. A method that takes this criterion explicitly into account is likely more suitable. Especially for subdivisions, continuity is probably unachievable unless vertices are allowed to move from their original position.

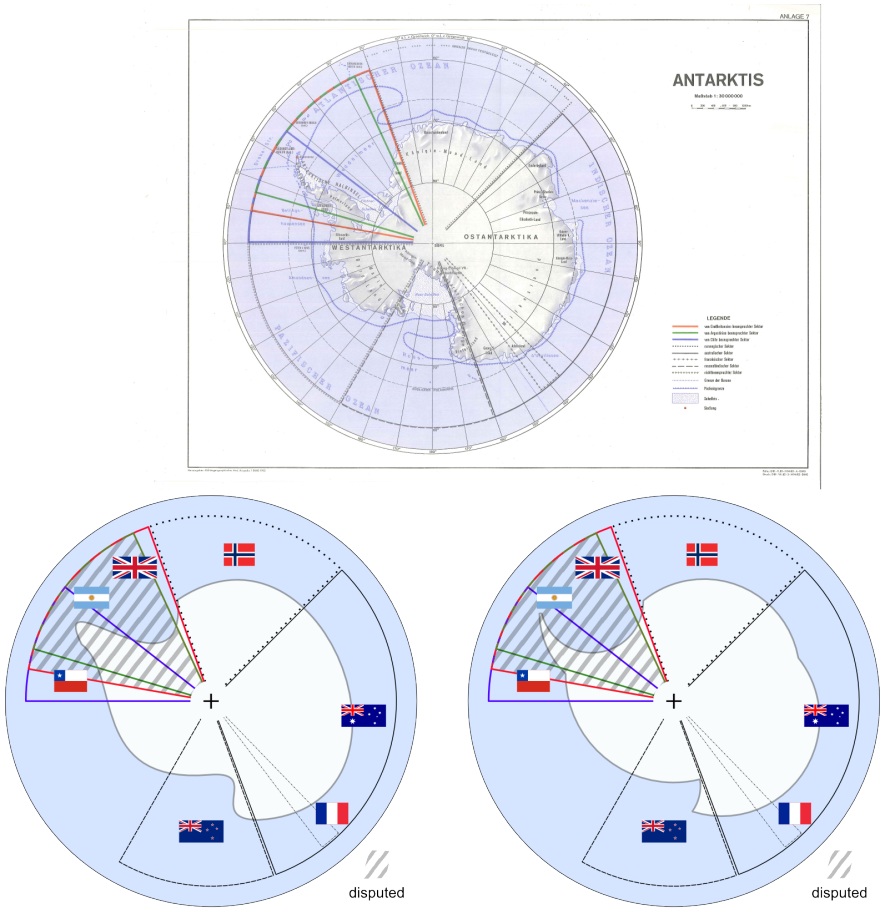

Figure 14: (Top) Downsized map of Antarctica (Militärgeographisches Amt (ed.) 1982). (Bottom) Diagrams using results of our framework.

References

Ahlberg, J., Nilson, E. & Walsh, J. (1967). *The Theory of Splines and Their Applications*, Academic Press.

Brunet, R. (1991). La population du Languedoc-Roussillon en 1990 et la croissance récente, *MappeMonde* **91**(1): 34–36.

Brunet, R. (2004). La Corse, région d'Europe, *Mappemonde* **76**(4): 1–16.

Bucher, H. & Schlömer, C. (2006). Die neue Raumordnungsprognose des BBR, *Raumforschung und Raumordnung* **64**(3): 206–212.

Buchin, K., Meulemans, W. & Speckmann, B. (2011). A new method for subdivision simplification with applications to urban-area generalization, *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 261–270.

Cicerone, S. & Cermignani, M. (2012). Fast and Simple Approach for Polygon Schematization, *Proceedings of the 12th International Conference on Computational Science and Its Applications*, LNCS 7333, pp. 267–279.

De Berg, M., Cheong, O., Van Kreveld, M. & Overmars, M. (2008). *Computational geometry: algorithms and applications*, 3rd edn, Springer.

De Casteljau, P. (1959). Outillages méthodes calcul, *Technical report*, A. Citroën.

De Chiara, D., Del Fatto, V., Laurini, R., Sebillo, M. & Vitiello, G. (2011). A chorem-based approach for visually analyzing spatial data, *Journal of Visual Languages and Computing* **22**(3): 173–193.

Drysdale, R., Rote, G. & Sturm, A. (2008). Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs, *Computational Geometry: Theory and Applications* **41**(1-2): 31–47.

Duncan, C., Eppstein, D., Goodrich, M., Kobourov, S. & Löffler, M. (2012). Planar and poly-arc Lombardi drawings, *Graph Drawing*, LNCS 7034, pp. 308–319.

Guilbert, E. & Lin, H. (2007). Isobathymetric line simplification with conflict removal based on a B-spline snake model, *Marine Geodesy* **30**(1-2): 169–195.

Gürtler, A. (1929). *Das Zeichnen im erdkundlichen Unterricht. Zweites Heft: Europa*, Wunderlich.

Heimlich, M. & Held, M. (2008). Biarc approximation, simplification and smoothing of polygonal curves by means of Voronoi-based tolerance bands, *International Journal of Computational Geometry and Applications* **18**(3): 221–250.

Lee, D. & Drysdale, R. (1981). Generalization of Voronoi diagrams, *SIAM Journal on Computing* **10**(1): 73–87.

Masood, A. & Ejaz, S. (2010). An efficient algorithm for robust curve fitting using cubic Bezier curves, *Advanced Intelligent Computing Theories and Applications*, LNCS 6216, pp. 255–262.

Mi, X., DeCarlo, D. & Stone, M. (2009). Abstraction of 2D shapes in terms of parts, *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '09, pp. 15–24.

Militärgeographisches Amt (ed.) (1982). Antarktis 1:30.000.000, *Atlas der militärischen Landeskunde DMG-1982* .

Mustafa, N., Koutsofios, E., Krishnan, S. & Venkatasubramanian, S. (2001). Hardware-assisted view-dependent map simplification, *Proceedings of the 17th annual Symposium on Computational Geometry*, pp. 50–59.

Piegl, L. & Tiller, W. (1997). *The NURBS book*, Springer.

Regnauld, N. & McMaster, R. (2007). A Synoptic View of Generalisation Operators, *Generalisation of geographic information: cartographic modelling and applications*, Elsevier, pp. 37–66.

Reimer, A. (2010). Understanding chorematic diagrams: towards a taxonomy, *The Cartographic Journal* **47**(4): 330–350.

Reimer, A. & Meulemans, W. (2011). Parallelity in chorematic territorial outlines, *Proceedings of the 14th ICA Workshop on Generalization and Multiple Representation*.

Roberts, M. (2012). *Underground maps unravelled - Explorations in information design*.

Rote, G. (2007). Computing the Fréchet distance between piecewise smooth curves, *Computational Geometry: Theory and Applications* **37**(3): 162–174.

Saux, E. & Daniel, M. (1999). Data reduction of polygonal curves using B-splines, *Computer-Aided Design* **31**(8): 507–515.

Schneider, P. (1990). An algorithm for automatically fitting digitized curves, *Graphic Gems*, Academic Press Professional, pp. 612–626.

Shao, L. & Zhou, H. (1996). Curve fitting with Bezier cubics, *Graphical models and image processing* **58**(3): 223–232.

Stone, M. & DeRose, T. (1989). A geometric characterization of parametric cubic curves, *ACM Transactions on Graphics* **8**(3): 147–163.

Töpfer, F. & Pillewizer, W. (1966). The principles of selection, *The Cartographic Journal* **3**(1): 10–16.

Van der Poorten, P. & Jones, C. (2002). Characterisation and generalisation of cartographic lines using Delaunay triangulation, *International Journal of Geographical Information Science* **16**(8): 773–794.