Topologically safe curved schematization

Topologically Safe Curved Schematization[*]

Arthur van Goethem[†] Herman Haverkort[†] Wouter Meulemans[†]
Andreas Reimer[‡] 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. Our fully-automated approach does not need critical points or salient features. We illustrate our framework with Bézier curves and circular arcs.

1 Introduction

A schematized map uses shapes of low complexity to represent geographic objects. So far, most research on automated schematization has concentrated on straight line segments with restrictions on the admissible directions. Many manually produced schematized maps, however, make use of curves [12]. Curves have greater expressive power than line segments: several line segments can often be replaced by a single, low-degree curve. This can make it easier to interpret maps (see Fig. 1).

When using curves in cartography we need topologically sound results. Generalization and other geoprocessing operations are often implemented in recursive processes that repeatedly make expensive checks for topology violations. Removing detected intersections is a challenge as well, triggering more checks for curve-curve intersections. To the best of our knowledge, our framework is the first that ensures topology preservation for curved schematization operations while avoiding repeated checks for intersections.

Contribution. We describe a generic framework for computing curved schematizations of simple polygons that maintain the topology of the input. Our framework segments the input polygon at a subset of its vertices and fits curves to the resulting polygonal *chains*,

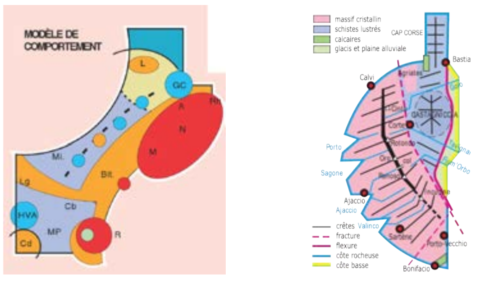

Figure 1: Languedoc-Roussillon [1], Corsica [2].

which consist of one or more (consecutive) edges of the input polygon. 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.

To use the framework two components need to be specified. The first is a method to fit a simple curve to a polygonal chain. The second is a distance measure which expresses how well the curve fits. There are two requirements for 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. 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. Straight lines are also curves and could be used in our framework, resulting in "classic" simplification.

The quality of the schematization depends on the segmentation. We formulate the problem of finding the best segmentation as an optimization problem (Section 2). Hence we do not need critical points or salient features to be specified by the user. To illustrate our framework we use Bézier curves (Section 3) and area-preserving circular arcs (Section 4) to generate curved schematizations of territorial outlines.

Related work. Automated simplification and smoothing with straight lines have received significant attention over the years. Mustafa *et al.*[11] use Voronoi cells for topologically safe simplification and heuristics to speed up the process. Our framework with straight lines and the directed Hausdorff distance yields a similar setting. Van der Poorten and Jones [18] use constrained Delaunay triangulations to find and simplify features in a topologically safe way.

A variety of methods fit a (cubic) Bézier curve to

[*]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).

[†]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

a polygonal line [9, 15, 16]. 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 [15] applies a heuristic similar to Douglas-Peucker to fit multiple curves, in essence using critical points. Shao and Zhou [16] also use critical point detection before fitting Bézier curves. B-splines have been applied to approximate polygons [7, 14]. Intersections of different splines, however, still need to be checked and resolved separately. Existing work on automated schematization of shapes is sparse. So far it has concentrated on straight-line drawings.

Drysdale *et al.* [5] 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 [8] 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 Computing optimal segmentations

As stated earlier, the quality of the final schematization depends on the segmentation. In this section we describe how to efficiently use dynamic programming to find an optimal segmentation.

We first compute the edge-based Voronoi diagram of the input polygon. Every Voronoi cell is a simple polygon (potentially unbounded and with parabolic parts) without holes. The Voronoi cell of a chain is the union of the Voronoi cells of its edges. To guarantee topological correctness, we ensure that the union of cells remains a simple polygon without holes (see Fig. 2).

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. For $i = j$, this is the entire polygon. Each entry $T[i, j]$ in the table will contain the curve fitted to $P[i, j]$ and a value that indicates the quality of the fit. For each entry of T

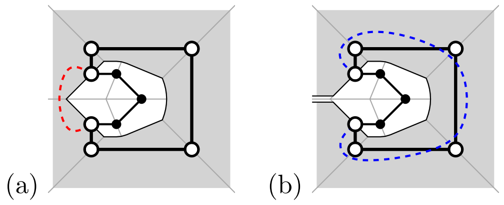

Figure 2: (a) Dashed curve lies in the cell but violates topology. (b) Union considers the pocket.

we compute the associated cell 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 by setting $T[i, j] = \infty$ to ensure topological correctness.

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 $O(n^2F(n))$ time to compute the lookup table.

To obtain a schematization from the table T, we select the curves representing a collection of consecutive chains which 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 has 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 $k = K$:

$$OPT[i, k] =$$

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

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).

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 has 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, this may create visually salient points not present in the input data (see Fig. 3a). To avoid such visual artifacts we use a weighted average 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. 3b). We deem the disappearance of features less disturbing than the appearance of features that do not exist.

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 [17].

Fitting a cubic Bézier curve. For fixed tangents at the endpoints, Schneider [15] shows how to algebraically compute the Bézier curve optimizing

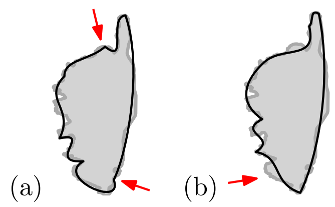

Figure 3: (a) Regular squared error. (b) Weighted squared error.

the least-squares measure in linear time. A similar method can be used to optimize a weighted measure. We direct the initial tangents towards the furthest vertex on either side of the straight line connecting the endpoints (see [9]). Based on these tangents, the algorithm by Schneider optimizes the distance of the second and third control point. We subsequently fix the distance between control points P_0 and P_1, and between P_2 and P_3 and optimize the tangents for either side algebraically. If desired, this process of subsequently 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. If required the curve can be subdivided up to a constant number of times using De Casteljau's algorithm [4] 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 total preprocessing time within our framework is $O(n^3)$.

Results. Fig. 4 and 5 show results of our framework. Each depicts a schematization of different complexity on top of the input polygon. Even a small number of curves result in recognizable and pleasing shapes.

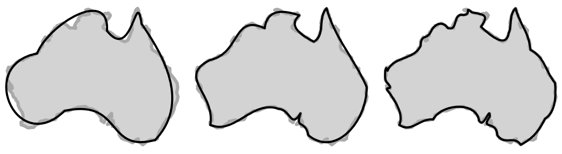
Figure 4: Australia with 5, 10, and 20 curves.

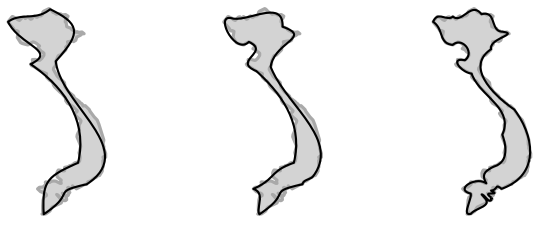
Figure 5: Vietnam with 7, 10, and 20 curves.

4 Circular arcs

Inspired by recent work on area-preserving schematization [3] and circular-arc graph drawing [6], we use our framework for area-preserving circular-arc schematizations. Given a chain, the circular arc with the same start and end vertex that preserves the area on each side of it is unique. It is computed by fitting a circular segment with a signed area equal to that of the chain. This requires numerical methods. As distance measure we use the continuous Fréchet distance, extended for curves [13]. We find that $F(n) = O(n \log n)$ and thus, precomputing the lookup table requires $O(n^3 \log n)$ time.

Results. Fig. 6 and 7 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. A few more arcs give a stylized or playful appearance and make the shape more recognizable.

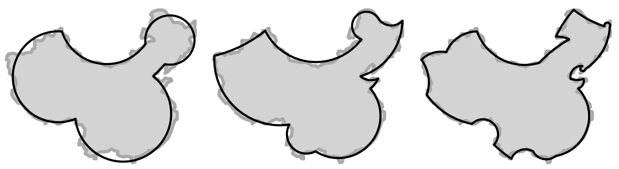
Figure 6: China with 5, 10, and 20 arcs.

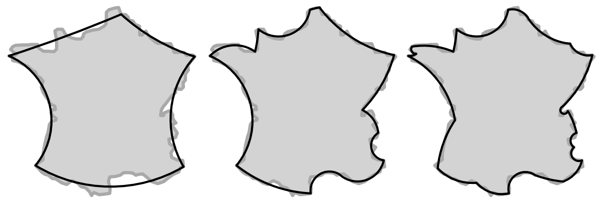
Figure 7: France with 5, 12, and 20 arcs.

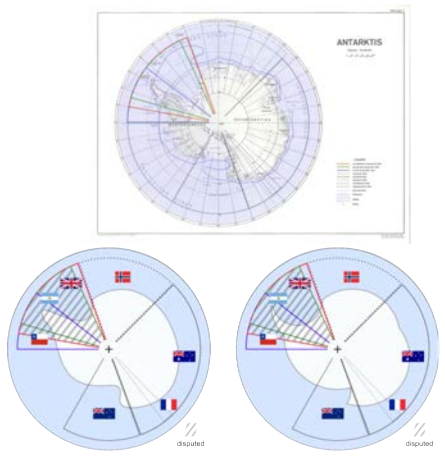

Figure 8: (Top) Downsized map of Antarctica [10]. (Bottom) Diagrams using results of our framework.

References

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

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

[3] K. Buchin, W. Meulemans, and B. Speckmann. A new method for subdivision simplification with applications to urban-area generalization. In *Proc 19th ACM GIS*, pages 261–270, 2011.

[4] P. De Casteljau. Outillages méthodes calcul. *Technical report, A. Citroën*, 1959.

[5] R. Drysdale, G. Rote, and A. Sturm. Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs. *CGTA*, 41(1-2):31–47, 2008.

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

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

[8] M. Heimlich and M. Held. Biarc approximation, simplification and smoothing of polygonal curves by means of Voronoi-based tolerance bands. *IJCGA*, 18(3):221–250, 2008.

[9] A. Masood and S. Ejaz. An efficient algorithm for robust curve fitting using cubic Bezier curves. In *Proc 6th ICIC (LNAI 6216)*, pages 255–262, 2010.

[10] Militärgeographisches Amt (ed.). Antarktis 1:30.000.000. *Atlas der militärischen landeskunde DMG-1982*, 1982.

[11] N. Mustafa, E. Koutsofios, S. Krishnan, and S. Venkatasubramanian. Hardware-assisted view-dependent map simplification. In *Proc 17th SCG*, pages 50–59, 2001.

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

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

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

[15] P. J. Schneider. An algorithm for automatically fitting digitized curves. In *Graphic Gems*, pages 612–626. Academic Press Professional, 1990.

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

[17] M. Stone and T. DeRose. A geometric characterization of parametric cubic curves. *ACM Trans. on Graph.*, 8(3):147–163, 1989.

[18] P. van der Poorten and C. Jones. Characterisation and generalisation of cartographic lines using Delaunay triangulation. *International Journal of GIS*, 16(8):773–794, 2002.