Scalable Harmonious Simplification of Isolines

Steven van den Broek
TU Eindhoven, the Netherlands

Wouter Meulemans
TU Eindhoven, the Netherlands

Andreas Reimer
TU Eindhoven, the Netherlands
Arnold-Bode-Schule Kassel, Germany

Bettina Speckmann
TU Eindhoven, the Netherlands

Abstract

Isolines visually characterize scalar fields by connecting all points of the same value by a closed curve at repeated intervals. They work only as a set which gives the viewer an indication of the shape of the underlying field. Hence, when simplifying isolines it is important that the correspondence – the harmony – between adjacent isolines is preserved whenever it is present. The majority of state-of-the-art simplification methods treat isolines independently; at best they avoid collisions between adjacent simplified isolines. A notable exception is the work by Van Goethem et al. (2021) who were the first to introduce the concept of harmony between adjacent isolines explicitly as an algorithmic design principle. They presented a proof-of-concept algorithm that harmoniously simplifies a sequence of polylines. However, the sets of isolines of scalar fields, most notably terrain, consist of closed curves which are nested in arbitrarily complex ways and not of an ordered sequence of polylines. In this paper we significantly extend the work by Van Goethem et al. (2021) to capture harmony in general sets of isolines. Our new simplification algorithm can handle sets of isolines describing arbitrary scalar fields and is more efficient, allowing us to harmoniously simplify terrain with hundreds of thousands of vertices. We experimentally compare our method to the results of Van Goethem et al. (2021) on bundles of isolines and to general simplification methods on isolines extracted from DEMs of Antartica. Our results indicate that our method efficiently preserves the harmony in the simplified maps, which are thereby less noisy, cartographically more meaningful, and easier to read.

2012 ACM Subject Classification Information systems → Geographic information systems; Theory of computation → Computational geometry

Keywords and phrases Simplification, isolines, harmony

Digital Object Identifier 10.4230/LIPIcs.COSIT.2024.8

Supplementary Material *Software (Source code)*: `https://github.com/tue-alga/cartocrow` archived at `swh:1:dir:f600a45ca2f26faaf6b06ca30c5da09cbd261e23`
Dataset: `https://doi.org/10.17605/OSF.IO/A67JP`

Funding *Wouter Meulemans*: Partially supported by the Dutch Research Council (NWO) under project number VI.Vidi.223.137.

1 Introduction

The goal of simplification is to represent shapes using less complex geometry, while maintaining important features. As a geometric form of data compression, it has numerous applications including efficient data structures [7] and approximation algorithms [19]. We focus on its cartographic applications: deriving less complex shapes in generalization [29] or schematization [8]. Here simplification improves the quality of scale-appropriate maps and

eliminates unnecessary detail. A common approach to simplification is to first establish a quality measure to quantify the geometric distortion between the original and its simplified shape. This measure is subsequently optimized – often heuristically – to obtain the simplification. Such methods frequently yield high-quality simplifications when shapes are treated in isolation. However, they tend to consider other shapes in the neighborhood less as context and more as obstacles that need to be handled appropriately to avoid intersections or other topological distortions. Hence, where a composition of shapes may display common features, individually simplified versions may become discordant and lose their common ground.

Van Goethem et al. [15] first studied the idea of *harmony*: harmonious simplification does not happen in isolation but strives to maintain common features as far as they are present in the input. This concept is particularly salient when simplifying sets of isolines: the isolines together describe common features such as hills and valleys. Where they describe a single smooth slope, this appearance should be maintained after simplification. Van Goethem et al. not only introduce the concept of harmony in simplification, but they also present an algorithm that computes harmonious simplifications. This algorithm is based on the rather restrictive concept of *locally-correct Fréchet matchings* [6] and can therefore be applied only to a sequence of polylines (for example, describing a single slope). To compute harmonious simplifications of general sets of isolines that are derived from arbitrary terrains (or more generally two-dimensional functions) we hence need another algorithmic approach.

Results and organization. We propose a new method for harmonious simplification that overcomes the drawbacks of locally-correct Fréchet matchings and can hence handle sets of isolines with hundreds of thousands of vertices derived from complex terrains. Our algorithm finds harmonious pieces within sets of isolines using the medial axis; this allows us to treat arbitrary complex nesting patterns of isolines and also improves efficiency. Furthermore, we introduce new methods to simplify harmonious pieces of isolines; our methods are both less restrictive and more efficient than the corresponding operations used by Van Goethem et al.

In the following, we first briefly review the concept of harmony as discussed by Van Goethem et al.; please see their paper for a more extensive treatment. Next, we discuss related work. Since our algorithm follows the general outline of the approach by Van Goethem et al. (identify harmonious pieces of isolines and then iteratively simplify them) we dedicate the separate Section 1.2 to a description of their paper. In Section 2 we then explain our new algorithm in detail. We implemented our algorithm and compared it experimentally to both Van Goethem et al. (by necessity only on ordered sets of isolines) and to general simplification methods that can handle isolines derived from terrains of high complexity, such as the terrain of Antartica. In Section 3 we describe our experimental setup and discuss our results. These results show that our method preserves harmony at least as well and often better than Van Goethem et al. on sets of ordered isolines. On sets of isolines that stem from general terrains, our method maintains harmony whenever it is present in the input and produces results that are cleaner and easier to read than those of non-harmonious methods. Last but not least, our method computes harmonious simplifications for isolines with close to a million of vertices in less than half an hour.

Harmony. Isolines are a method to visually and cartographically depict a scalar field via repetition and change of (mostly closed) curves. Isolines work only as a set: from the underlying scalar field, a family of curves is derived that shows, by the dialectic of repetition and change in comparison to each other, how values of a third (or higher) dimension are distributed in the underlying space. This dialectic of repetition and change should follow the

aesthetic principle of harmony. The question then arises how this aesthetic principle might be measured to be controlled or optimized by an algorithm. To operationalize harmony into measurements, Van Goethem et al. [15] used the extant design rules for contour lines (isolines on cartographic terrains) discussed in the seminal works by Eduard Imhof. They then developed the algorithmic notion of harmony from a process suggested by Goldman [16] and applied to cartography by Reimer [30].

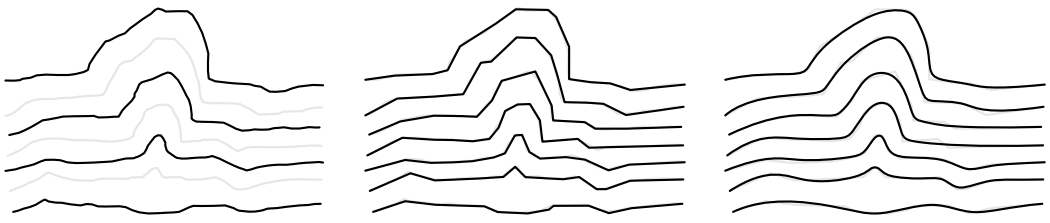
1.1 Related work

Our work falls squarely within the broad area of cartographic generalization. This field is concerned with re-modelling cartographic objects for various purposes such as data reduction, noise removal, and the optimization of visual communication such as the creation of scale-appropriate maps [33]. More precisely, the simplification of contour lines is a form of terrain generalization; Guilbert et al. [17] provide an introduction to the topic and a survey of generalization methods. The process of generalization can be decomposed into subprocesses called generalization operators [29, 33]. A selection of operators we deem important in the context of contour line generalization are given below, see also Figure 1.

Selection. Selection of objects; in our case contour lines.
Simplification. Reduction of the data used to represent an object.
Smoothing. Removal of details and perturbations in objects.

The algorithm we present in this paper is a simplification operator. Methods for terrain simplification can be categorized into two fundamentally different approaches: line simplification and surface simplification [18, 27]. Line simplification simplifies the contour lines directly, whereas surface simplification first simplifies the underlying terrain model (usually a triangulated irregular network (TIN)), and recomputes contour lines based on the simplified terrain model. Both techniques have advantages and disadvantages. Surface simplification does not handle isolines directly, which prevents the direct application of cartographic techniques. Furthermore, additional data, such as drainage networks, need to be re-derived from the simplified TIN as well, to match the simplified terrain. The major problem of line simplification approaches is the fact that common features of the underlying surface are generally not simplified together and existing harmony between the isolines is not preserved. We present a line simplification technique that tackles this problem.

A common approach to line simplification is the iterative removal of points. An example of such an algorithm is the well-known Ramer-Douglas-Peucker algorithm [10, 28] that is widely used in GIS. When such algorithms are applied on each isoline in isolation, the topological relation of curves may change; in particular, simplified isolines may intersect. Dyken et al. [11] present a vertex-removal algorithm that uses a constrained Delaunay triangulation to



(a) Selection **(b)** Simplification **(c)** Smoothing

Figure 1 Generalization operators. Isolines from Imhof [22] digitized by Van Goethem et al. [15].

efficiently determine whether a removal would change the topology and does not execute any topology-changing vertex removals. More recently, various algorithms have been introduced that iteratively replace edges with a single point, so-called edge *collapses* [15, 25, 32]. This approach is more flexible than the vertex removal approach as the simplification may use vertices that are not part of the input. A criterion often used in combination with edge collapses is area preservation [5, 8, 15, 25, 26, 31], which requires the areas of the geometry to be preserved. Area-preserving simplification tends to be of high visual quality and also comparatively efficient, since the search space for new vertex positions is restricted.

Van Goethem et al. [14, 15] were the first to explicitly simplify shapes simultaneously to maintain common features and promote parallelism. Their first paper [14] describes a vertex-removal simplification algorithm that optimizes parallelism according to some predefined measure; their second paper [15] introduces an algorithm that applies area-preserving edge collapses simultaneously to preserve harmony. We describe this algorithm in more detail below. Though both papers present conceptually strong results, the algorithmic side is limited to proof-of-concept methods that are not sufficiently general and efficient for practical use. Our algorithm builds on their ideas but introduces novel approaches for all parts to be able to handle sets of general isolines with hundreds of thousands of vertices.

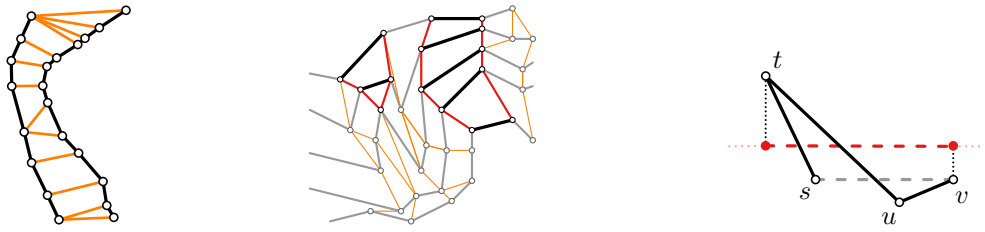
1.2 Harmonious simplification of isolines

We now briefly summarize the algorithm by Van Goethem et al. [15]; we refer the reader to the original work for further details. The algorithm has two stages; we use the same structure in our adaption. Recall that the input is a sequence of ordered polylines.

Stage 1: slope skeleton. In the first stage, consecutive polylines are matched. The matchings are required to be vertex-based and non-decreasing: each vertex in the one polyline is matched to at least one vertex in the other and matched pairs of vertices respect the order in which they occur along their respective polylines. This is achieved by using locally correct Fréchet matchings with the geodesic distance (Figure 2a).

These matchings together define a *slope skeleton* (Figure 2b), which partitions the space between the polylines into quadrilaterals and triangles. A *slope ladder* is a maximal set of such triangles and quadrilaterals, which is created by combining those triangles and quadrilaterals that share an edge of a polyline. Note that triangles can appear only at the ends of a slope ladder. The polyline edges insides a slope ladder are its *rungs*.

Stage 2: iterative reduction. In the second stage, the complexity of the polylines is gradually reduced by iteratively collapsing slope ladders. Such a collapse replaces each rung



(a) Locally correct Fréchet matching

(b) Slope skeleton (orange), highlighted slope ladders (red), and rungs (black)

(c) Area-equivalence line (red) and span (thick dashed red) of rung tu

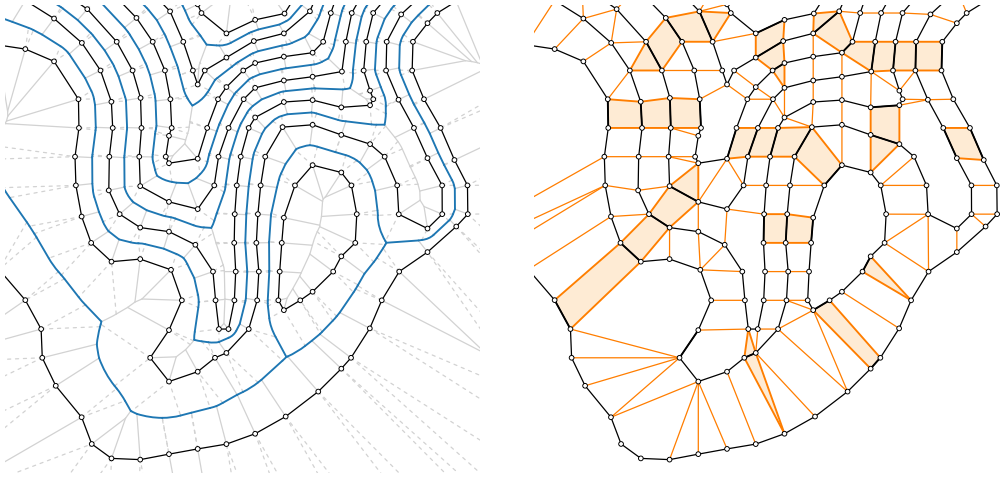
Figure 2 Figures from Van Goethem et al. [15] illustrating several definitions.

of a ladder by a single vertex in an area-preserving manner. Hence, the new vertex must lie on a specific line: the *area-equivalence line* (Figure 2c). The new positions are further restricted to lie within a specific *span* of this line, determined by projecting the rung and the edge before and after onto the area-equivalence line. Finally, the new vertices within these spans are determined by the so-called *harmony line* that connects the midpoint of the first rung to the midpoint of the last rung of the ladder. Typically the new vertex replacing a rung is the intersection between the corresponding area-equivalence line and the harmony line, but if this intersection lies outside of the span, it is the closer endpoint of the span instead. The algorithm considers not only one but many parallel copies of the harmony line. The final harmony line and corresponding new vertices are chosen such that the directed Hausdorff distance from the new edges to the corresponding part of the original polyline is minimized. The algorithm collapses only slope ladders that cause no topological changes. Among those, it repeatedly selects the slope ladder whose collapse causes the least distortion, as measured via the average symmetric difference (areal displacement) caused by each collapsed rung. The overall running time is $O(n^2 \log n)$, where n is the total number of vertices in the polylines.

In contrast, our algorithm runs in $O(k_{max} \cdot n \log n)$ expected time, with k_{max} denoting the length of the longest slope ladder, under some standard input assumptions that we discuss in detail in the next section. Note that this is a somewhat loose upper bound: the factor k_{max} contributes only when there are many long slope ladders that cause topological changes that one cannot collapse and the slope ladders that one does collapse are short.

2 Algorithm description

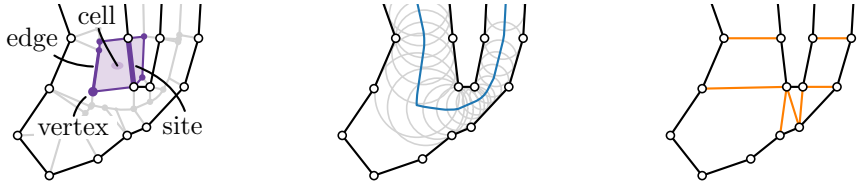
At the core of our algorithm lies the identification of slope ladders (bundles of edges spanning across isolines) and their simultaneous collapse. Slope ladders arise from a matching between the vertices of the isolines (Figure 3b). We employ a subset of the medial axis [3] (Figure 3a) to both identify which parts of isolines to match and to construct a matching. In Section 2.1 we describe how our matching is constructed, and we proceed in Section 2.2 with a description of how these matches are linked to form the slope ladders. After these preprocessing steps, the algorithm simplifies isolines in an iterative manner by selecting and collapsing a slope



(a) Voronoi diagram, medial axis (solid) and medial axis separator (blue)

(b) Matching (orange), highlighted slope ladders (thick, filled) and their rungs (thick, black)

Figure 3 Medial axis and derived slope ladders.



(a) Voronoi diagram **(b)** Medial axis separator **(c)** Matching

Figure 4 Computing a matching.

ladder in each iteration. Section 2.3 describes how a single collapse is performed and how the slope ladders and related data structures are updated after such a collapse. Finally, in Section 2.4, we explain how the algorithm chooses which slope ladders to collapse, and give an overview of the entire algorithm, including an analysis of the running time.

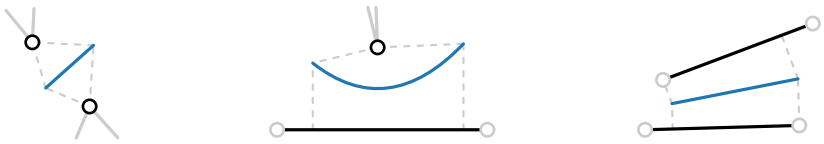
2.1 Preprocessing: matching isolines

Intuitively, one can think of our approach as follows. Imagine the isolines as physical barriers. Then move a disk – spring-loaded, such that it is always of maximal size – through the regions between the isolines (Figure 4b). Parts of distinct isolines touched by the disk are matched to each other (Figure 4c). This idea is captured formally by the *medial axis* [3] of a shape S, which is the set of centers of disks of maximum radius that touch S in at least two points. As we are interested in matching distinct isolines, we use the subset of the medial axis that consists of disks of maximum radius that touch at least two different isolines. We refer to this subset as the *medial axis separator*, following Bereg [1]. See Figure 4b for an illustration of the medial axis separator and a selection of disks of maximum radius.

To compute the medial axis separator, we first subdivide the plane into regions called *cells* that have the property that all points in the interior of a cell are closest to a unique *site*: a vertex or edge of an isoline. That is, we compute a *Voronoi diagram* [2] of the vertices and edges of the isolines (Figure 4a). Cells of the Voronoi diagram are separated by *Voronoi edges* which meet at *Voronoi vertices*. The two sites to which all points on a Voronoi edge are equidistant are called the *defining sites* of the Voronoi edge. The medial axis separator consists of all Voronoi edges whose defining sites belong to different isolines. Figure 3a illustrates the Voronoi diagram, the medial axis, and the medial axis separator.

Each Voronoi edge of the separator gives rise to a matching between its defining sites. Bereg [1] uses this fact to define a continuous monotone matching between two polylines. That is, any point on the polylines, including points on edges, are matched and the matched pairs of vertices respect the order in which they occur along the respective polylines. Our purposes require a monotone matching between vertices, which we obtain as follows.

There are three cases: the defining sites of a Voronoi edge may be two vertices, two line segments, or a vertex and a line segment (Figure 5). If the defining sites of a Voronoi edge



(a) Vertices **(b)** Vertex and line segment **(c)** Line segments

Figure 5 Types of edges of the medial axis separator (blue) and their defining sites (black).

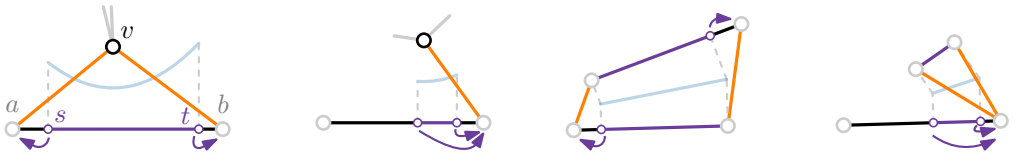


Figure 6 Examples of Voronoi edge projections with the resulting matchings.

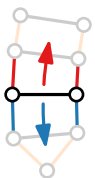
are two vertices then we simply match those two vertices. If one site is a line segment ab and the other site is a vertex v, then we first project the Voronoi edge on ab. Call the endpoints of the projection s and t (Figure 6, left). Determine for s and t whether they are closer to a or to b. If they are both closest to a then match a to v. Similarly if they are both closest to b then match b to v. If one is closest to a and one is closest to b, then match both a and b to v. If the defining sites of a Voronoi edge are line segments e_1 and e_2 then the same projection is performed, only now on both line segments. If each projected endpoint is closest to a unique line segment endpoint then the endpoints of e_1 are matched to the endpoints of e_2 respectively (Figure 6, middle right). Otherwise, the setting reduces to a case that has already been described. Thus three situations can occur: the endpoints of two segments are matched to each other; the endpoints of a segment match to a vertex; or one vertex matches to one other vertex. The union of these vertex matchings induced by all edges of the medial axis separator form our monotone matching between isolines.

We partition the vertex matchings of an isoline into two sets based on the side of the isoline on which the target of a matching lies. That is, we partition the matchings into a set to the left and to the right of the isoline (if the isoline is open) or into a set on the inside and the outside of the isoline (if the isoline is closed). This can be done efficiently during the creation of the matchings. By definition isolines are closed, but some may be cut off by the map boundary. The endpoints of such open isolines are not matched and the incident edges never collapsed. Therefore, these endpoints will always remain at their initial position.

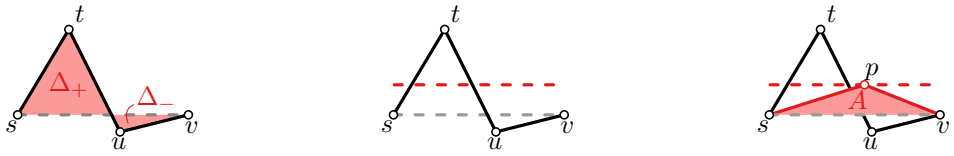
2.2 Preprocessing: deriving slope ladders

We link the vertex matchings of isolines to form slope ladders. We define a slope ladder as a sequence of harmonious edges spanning across distinct isolines, possibly with a vertex at the start of the sequence and possibly with a vertex at the end of the sequence. We refer to the edges in a slope ladder as *rungs* and the possible vertices at the end of the sequence as *caps*.

We say an edge ab of an isoline is harmonious with a vertex v of another isoline if a and b both match to vertex v. Similarly, we view an edge ab of an isoline as harmonious with an edge cd of another isoline if vertex a matches to c, but not to d, and vertex b matches to d, but not to c. There are a couple additional edge cases that we handle in our implementation, but we do not discuss them here as they require detailed explanation but are uninteresting.



We now construct slope ladders as follows. We iterate over all edges of isolines. If an edge e is not yet part of a slope ladder then we create a new slope ladder which initially contains only e as a rung. Note that by construction of the matching, edge e is harmonious with at most one edge or vertex on each side of its isoline. If there are such harmonious edges or vertices then we add them to the slope ladder. If a harmonious edge e' was found then recursively search for more harmonious edges or vertices on the other side of the isoline that e' belongs to.



(a) The area A contributed by $stuv$ equals $\Delta_+ - \Delta_-$

(b) Area-equivalence line (red)

(c) Area contributed by spv

Figure 7 Any point on the area-equivalence line creates a triangle that contributes the same area as before. Figures adapted from Figure 5 by Van Goethem et al. [15].

2.3 Iterative reduction: a single collapse

We simplify isolines iteratively by *collapsing* slope ladders. A slope ladder is collapsed by collapsing each of its rungs. That is, we replace each rung with a vertex. We consider only *area-preserving* collapses. For closed isolines, this means that the area of the region it encloses is preserved exactly. For open isolines, the preserved value is the signed area of the possibly self-intersecting polygon that is obtained by connecting the two ends of the isoline.

Area-preserving edge collapse. For an edge collapse to be area-preserving, the new vertex must lie on a line, called the *area-equivalence line*. For clarity we repeat the explanation by Van Goethem et al. [15]. Consider an edge tu with preceding and succeeding vertices s and v so that they are ordered s, t, u, v on the corresponding isoline. If an edge does not have preceding or succeeding vertices then we never collapse it. The area contributed by the edge tu and its adjacent edges is the signed area A of the polygon $stuv$ (Figure 7a). To preserve area, the new vertex p should be placed such that the signed area of triangle spv is equal to A. Taking sv as a fixed base, we see that the height of the triangle spv should be $h = 2A/|sv|$. Thus, the new vertex p is restricted to lie on a line parallel to sv at signed distance h to sv (Figure 7b). This line is referred to as the *area-equivalence line* for edge tu. Collapsing edge tu to any point p lying on this area-equivalence line preserves area (Figure 7c).

Ladder collapse. We collapse a ladder by performing area-preserving collapses of each of its rungs. Hence we have to choose points on the area-equivalence lines of each rung. Any type of area-preserving edge collapse can be used and applied independently on each rung. In our experiments we compare the two edge collapse methods described below.

Projected midpoint. Project the midpoint of tu on the area-preservation line.

Minimize areal displacement. Choose the point on the area-preservation line ℓ that minimizes the areal displacement of the edge collapse. Excluding degenerate cases, this is either the intersection between ℓ and the line through st, or the intersection between ℓ and the line through uv. This edge collapse was studied by Kronenfeld et al. [25]; see their paper for more details.

The simultaneous collapse of edges in a slope ladder already promotes harmony. However, this effect can be strengthened by coordinating the edge collapses of the rungs. Below, we describe the coordinated collapses we use in our experiments.

Line. As described in Section 1.2, Van Goethem et al. [15] use a harmony line to choose points on the area-equivalence lines. We are adapting their approach as follows. If a slope ladder consists of only one rung, then we apply the minimize areal displacement edge collapse. Furthermore, instead of minimizing the directed Hausdorff distance from the

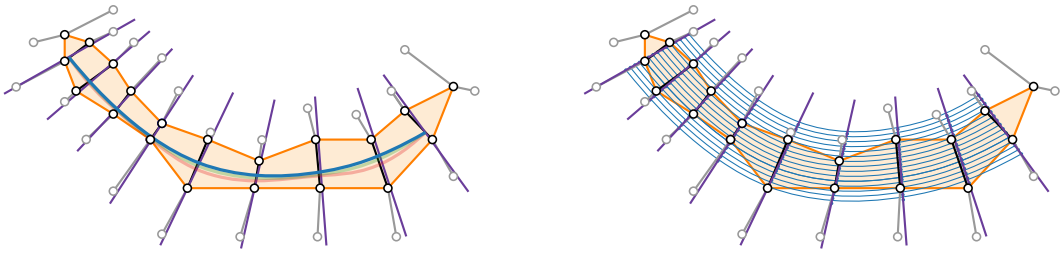


Figure 8 Illustration of the spline collapse, the area-equivalence line of each rung is indicated in purple. Left: the three spline iterations (red, green, blue); the blue spline is the final spline with minimum areal displacement (chosen from 15 samples). Right: the 15 samples.

newly inserted edges to the represented section of the original isoline, we minimize the areal displacement of the edge collapse instead. This makes the collapse more efficient and in our experiments did not result in any measurable or perceivable loss of quality.

Spline. The line collapse generally works well for small slope ladders that tend to be fairly straight. However, larger slope ladders may bend significantly and, therefore, a harmony line does not capture their shape well. The spline collapse uses intersections of the area-equivalence lines with a uniform B-spline as the new vertex positions. The initial harmony spline has the midpoints of the rungs as control points. Then we smooth the spline by iteratively creating new splines that have the would-be new vertex positions as control points (Figure 8). We choose to do this twice, though the number of repetitions can be set by the user as desired. Next, similar to the line method, offsets of the initial harmony spline are created and the one that results in the minimum areal displacement is chosen (Figure 8). Like our adapted line method, if a ladder consists of only one rung then we apply the minimize areal displacement edge collapse.

Hybrid. When a slope ladder is reasonably, but not completely, straight, the spline collapse puts vertices almost, but not quite, on a line. The hybrid method detects such cases and uses a harmony line instead. Specifically, it uses the harmony line method if the line through the midpoint of the first and last rung of the ladder intersects each rung of the ladder; otherwise, it uses the spline method.

Maintaining data structures. After a collapse, the isolines change and all data structures need to be updated appropriately. First, we update the Voronoi diagram. Next, any matchings and slope ladders that were derived from a part of the Voronoi diagram that changed are removed and recomputed.

2.4 Iterative reduction: overall algorithm

Selecting a collapse. Each iteration, we collapse the slope ladder whose collapse incurs the smallest *cost* defined as the areal displacement (symmetric difference) between the current isolines and those after collapse. The collapse of a slope ladder is not allowed if it changes the topology either by introducing intersections between isolines or by "moving over" another isoline such that it lies on a different side after collapse. We maintain the Voronoi diagram throughout our algorithm to efficiently detect changes in topology caused by a ladder collapse. Here it is helpful to view an edge collapse as removing three edges and two vertices of the isolines, and creating a new vertex along with two new incident edges. For our running time analysis we make the assumption that each new edge intersects a constant number of cells of

the Voronoi diagram in expectation. The details of the topology check are many, but not difficult; we refer the reader to the implementation of our algorithm.

To efficiently retrieve ladders of low cost we make use of a min-priority queue data structure (a min-heap). When retrieving a ladder the above mentioned topology check is performed. If a check is violated, we store the segment that caused the violation and assign a cost of infinity to the ladder collapse. After a collapse the algorithm checks whether any of the now removed segments was the cause of a topology violation; if so, then the corresponding ladder is assigned its proper cost again, and is a candidate again for collapse.

Running time analysis. Let n denote the number of vertices in the input. The Voronoi diagram can be created in expected $O(n \log n)$ time [4], and the matching and slope ladders can be derived from it in $O(n)$ time. Initializing the ladders with their collapses, and computing their cost, takes $O(n)$ time in total. Thus, preprocessing takes expected $O(n \log n)$ time. Next, we analyze the iterations. First, the slope ladder of minimum cost is retrieved and removed from the priority queue in $O(\log n)$ time. Determining whether its collapse changes the topology of the isolines takes expected $O(1)$ time for each rung of the ladder. Any ladder contains at most k_{max} rungs, where k_{max} is the length of the longest ladder in the input, which is bounded by the total number of isolines. Hence the topology check takes expected time $O(k_{max})$. If the topology changes then the ladder is assigned cost ∞ and moved to the back of the priority queue in $O(\log n)$ time. This process is repeated until a ladder is found whose collapse does not change the topology. We assume that in every iteration there are expected $O(1)$ ladders that violate a topology check. This is a reasonable assumption because these violations do not accumulate, but instead are checked only once for every ladder. Thus, selecting a ladder to collapse takes expected $O(k_{max} \log n)$ time. After a ladder collapse has been selected, it is executed. The Voronoi diagram is updated in expected $O(k_{max} \log n)$ time as $O(k_{max})$ updates are performed on the Voronoi diagram, each taking expected $O(\log n)$ time [9]. Other data is also updated in $O(k_{max})$ time. Thus in total the expected running time is $O(k_{max} \cdot n \log n)$.

3 Experimental setup and results

In this section we present the results of an implementation of our algorithm for various datasets. We compare our simplifications to the output of the harmonious algorithm by Van Goethem et al. [15] and two non-harmonious simplification methods: the vertex-removal algorithm by Dyken et al. [11] and the area-preserving edge collapse algorithm by Kronenfeld et al. [25]. Furthermore, we compare the effectiveness of the different slope ladder collapses described in Section 2.3. Finally, we experimentally evaluate the running time of our implementation on datasets of various sizes. Below we first describe our experimental setup and our datasets, then we perform a quantitative analysis of our results, which is followed by a qualitative analysis.

Experimental setup. Our implementation is written in C++ and makes use of the segment Voronoi Diagram in CGAL [23, 24]. The implementation is part of the CartoCrow framework that can be accessed at `https://github.com/tue-alga/cartocrow`. We use a target number of vertices as a stopping criteria for all simplification methods. Our method and that of Van Goethem et al. collapse a ladder each iteration; therefore, the number of vertices might decrease by more than one in each iteration. These algorithms are stopped once the number of vertices is at or below the target number of vertices. Unless noted otherwise, all

our figures are generated using the hybrid collapse that uses fifteen samples for generating line and spline offsets, and two smoothing iterations for the spline.

We generate figures for the method of Dyken et al. [11] using its implementation in CGAL [12]. Dyken et al. [11] describe three error measures. We use their hybrid distance measure that applies a penalty when the simplification step would cause isolines to come closer than a distance R (a parameter). We indicate this parameter with two blue lines: parameter R is the distance between the two lines. Simplifications with the method described by Kronenfeld et al. [25] are generated with our implementation by creating no matching or slope ladders and using the minimize areal displacement edge collapse.

Data. The authors of Van Goethem et al. [15] shared their original figures and data with us so we could perform a direct comparison (see Figures 12, 13, and 14). These include digitized versions of examples by Imhof [22], a digitized set of contour lines for the whole of Antarctica at 500m contour intervals (Figure 13), and isolines generated with QGIS from Version 1 REMA DEMs (various resolutions and equidistances) [21]. Furthermore, we generated new isolines from DEMs from The Reference Elevation Model of Antarctica (REMA). These DEMs are provided by the Byrd Polar and Climate Research Center and the Polar Geospatial Center under NSF-OPP awards 1043681, 1542736, 1543501, 1559691, 1810976, and 2129685. In particular, Figures 16, 17 and 18 were generated from REMA Mosaic Version 2.0 from Thurston island off the coasts of Ellsworth Land and Mary Byrd Land in 10m resolution [20]. We name the four largest datasets we discuss: **MSY** (7995 vertices, Figure 14), **LWI** (139,658 vertices, Figure 17), **SWI** (131,072 vertices), and **MTR** (369,977 vertices). The data is available at doi:10.17605/OSF.IO/A67JP.

3.1 Quantitative evaluation

To quantitatively compare the results of our algorithm to both harmonious and non-harmonious simplification algorithms we use two types of quality measures: one for measuring the similarity of the simplified map to the original input and two for measuring harmony.

Measuring similarity. We measure similarity by computing for each input isoline the symmetric difference with the simplified version and summing the values. We use the Boolean operations of the CGAL library [13] to implement this. In order to compute the symmetric difference for open isolines, we first close both isolines in a consistent manner. We normalize the symmetric difference by dividing it by the area of the bounding box of the input isolines.

Measuring harmony. We use two measures for harmony. The first measure is vertex alignment, which we adapt from the harmony measure used by Van Goethem et al. [15]. To measure the alignment of two vertices u and v on distinct isolines, we determine the angular bisectors $n(u)$ and $n(v)$ of their respective incident edges. The alignment between the two vertices is then the sum of the appropriate angle $\alpha(u)$ between $n(u)$ and uv and the angle $\alpha(v)$ between $n(v)$ and vu (Figure 9). Our measure differs slightly from that used by Van Goethem et al. since they choose $\alpha(v)$ and $\alpha(u)$ to be the smallest angle between the line uv and $n(u)$ which means the worst alignment is 180° as illustrated at the right of Figure 9. Intuitively, however, the worst alignment approaches 360° when the normals $n(u)$ and $n(v)$ point in the opposite direction. This measure is shown in Figure 10 in the form of the average vertex alignment in radians of all vertices matched by the matching explained in Section 2.1.

Our second measure for harmony is the number of slope ladders within a set of isolines as created by our preprocessing step (see Section 2.1 and 2.2). Since we can create a set of

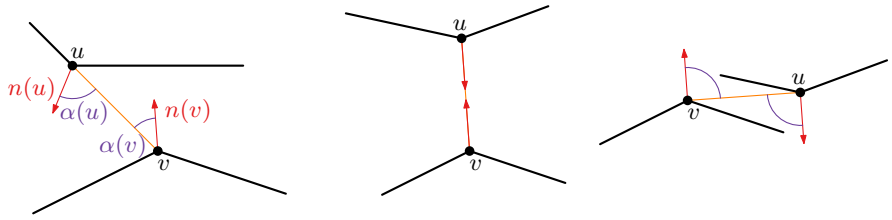


Figure 9 Figure from Van Goethem et al. [15] illustrating the vertex alignment measure. Left: the alignment of u and v is $\alpha(u) + \alpha(v)$. Middle: optimal alignment. Right: bad alignment of 180°.

slope ladders for any set of isolines, also those produced by a simplification algorithm that does not use slope ladders, we can use this measure across all outputs. When computing this measure for a set of isolines, we normalize it by dividing the number of slope ladders by the total number of vertices in the set of isolines.

Quantitative comparison. Figure 10 shows a comparison of different ladder collapse methods used within our algorithm and other simplification methods on four reasonably large datasets. An open symbol (and 1 after name) indicates a simplification method that does not use slope ladders. A filled symbol (and 2 after name) indicates a simplification method that does use slope ladders. We compare three different collapses: hybrid (Hybrid), the projected midpoint (Midpoint) and the minimize areal displacement (SymDiff). Note that the SymDiff collapse without slope ladders (SymDiff1) is the same method as the one described by Kronenfeld et al. [25]. In addition to these area-preserving edge collapse methods, we also measure the

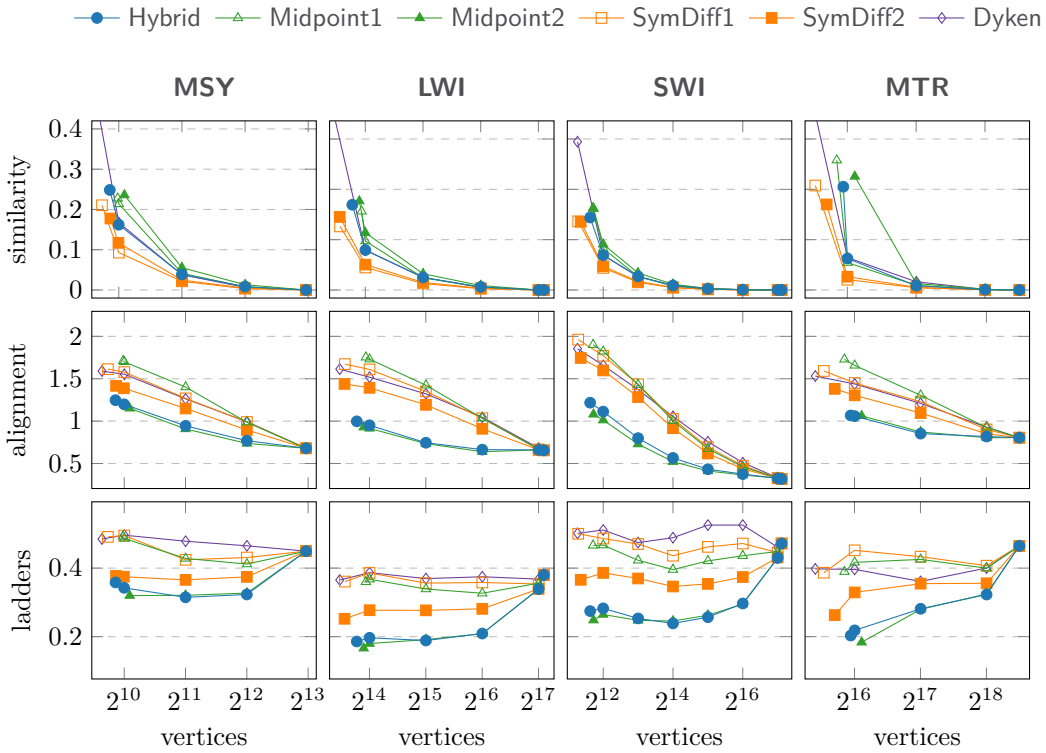


Figure 10 Isoline simplifications; lower values are better and the horizontal axis is logarithmic.

vertex-removal algorithm by Dyken et al. [11] (Dyken). In each subplot, at the far right is the input. The similarity is zero here. Then, the isolines are progressively simplified; harmony and similarity is measured when the number of vertices first drops below a power of two. When there is no longer a simplification step that preserves the topology of the isolines, the simplification stops. The measures at those times can be seen on the left of each subplot.

We now interpret the plot. Observe that in terms of harmony, any method that uses slope ladders outperforms any method that does not use slope ladders. Furthermore, introducing slope ladders for the Midpoint and SymDiff edge collapses significantly improves their score in terms of harmony. Most striking is the effect of slope ladders on the Midpoint collapse. Midpoint1 performs worst on vertex alignment, but Midpoint2 is tied for the best score on vertex alignment. The good performance of Midpoint2 on harmony is likely a result of the regular sampling of the DEMs which creates ladders with rungs of roughly the same length. Projecting the midpoint of the rungs on the area-preservation line yields aligned vertices, which also generally leads to fewer slope ladders. The SymDiff collapse scores best on similarity, which is expected as similarity is measured in terms of symmetric difference which is exactly what it minimizes. When this collapse is used in combination with slope ladders, the harmony improves, but the similarity measure deteriorates slightly. This is expected, as slope ladders restrict which edge collapses are performed. The Hybrid collapse performs best overall, as it is tied with Midpoint2 for first place in terms of harmony and outperforms Midpoint2 on similarity.

Experimental running time. We ran our experiments on a laptop with an 11th Gen Intel(R) Core(TM) i7-11800H 2.30GHz CPU and 32 GB, 3200 MHz, RAM. The results are averaged over ten runs. A dataset of close to a million vertices is included in the running time experiments, which is not present in Figure 10, due to artifacts (missing data) that prevent measurement of similarity. Figure 11 shows the results. The left plot shows the time in milliseconds spent per simplified vertex, as a function of the input size. This includes the time spent for preprocessing. The right plot shows the total running time needed to simplify the various datasets until no simplification step exists that preserves topology. Observe that isolines consisting of up to a million vertices can readily be simplified in less than half an hour. Furthermore, the running time increases slowly as the number of vertices in the input increases, and matches the running time one expects based on the theoretical analysis.

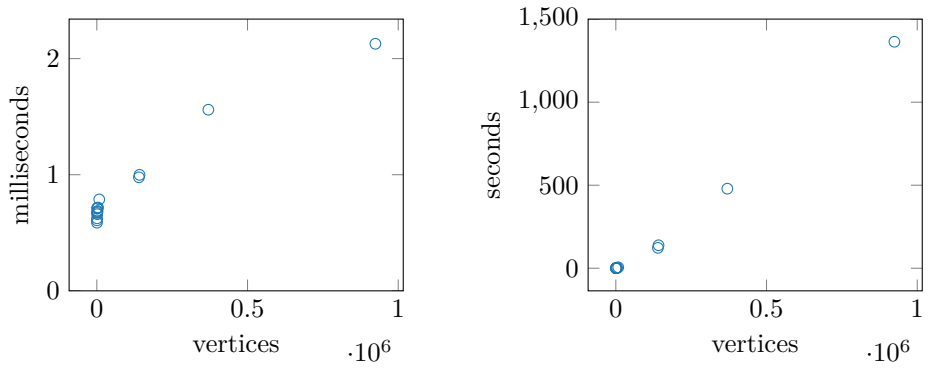


Figure 11 Left: time spent per simplified vertex as a function of the input size. Right: total running time as a function of the input size. Results are averaged over ten runs.

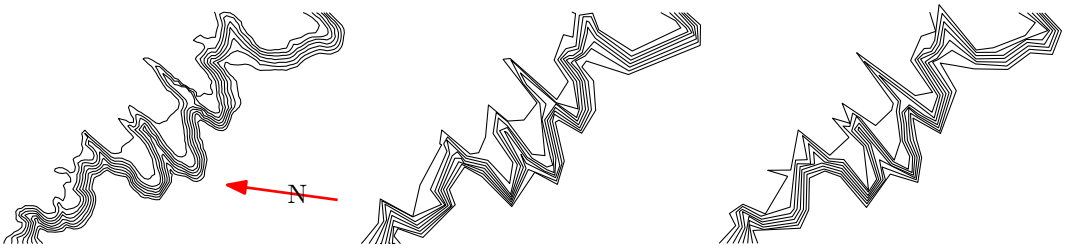


Figure 12 Left: input isolines consisting of 2071 vertices. Middle: Van Goethem et al., 246 vertices. Right: our method with hybrid collapse, 245 vertices.

3.2 Qualitative discussion

In this section we visually compare the results of the simplification approaches which we evaluated quantitatively in the previous section. The contour lines in the datasets we use for our comparison are already at an appropriate level of detail for the corresponding contour interval. As a result, the simplifications we show may appear coarse and are not necessarily scale-appropriate. However, they do support visual comparison well, since the contour lines are generally quite challenging to simplify in a topologically sound manner and hence readily reveal the differences between simplification algorithms. We include one figure (Figure 15) that is more scale-appropriate and ask the reader to use a digital zoom to inspect details.

Figure 12 highlights several key differences between our approach and Van Goethem et al., which we feel are improvements. A basic principle of generalization is to keep and possibly even reinforce recognizable shapes while reducing extraneous detail at the same time. Our hybrid approach does exactly that. Consider the south-eastern (top-right) slopes: where Van Goethem et al. straightens out the slope into a straight northwest-southeast direction, our method keeps the S-form of the input. Furthermore, our approach keeps the little sub-form marked by the discrepancy between the lowest and second lowest isoline. The narrow lobe is emphasized and harmonised to longer, parallel segments than the more winding input. The next spur ends in a double-pronged form, that is kept in our approach, but lost in the middle figure. We conclude that on this dataset our approach improves both the harmony between the isolines and keeps more characteristic shapes, while using the same number of vertices.

The results in Figure 13 look quite similar on first sight. A detailed look reveals some artifacts of Van Goethem et al.: over-simplification of the shallow western peninsula and extraneous vertices, for example, at the lowest isoline, breaking harmony without adding any characteristic shape. Also the inlet on the upper right shows a discontinuity in the isolines. However, in some other places, such as the upper bundle of isolines, our hybrid approach does not align vertices perfectly on a line, while Van Goethem et al. does. This difference can be attributed to the difference in matchings used by the algorithms. Empirically we have observed that the medial axis matching creates shorter slope ladders. This leads to less simultaneous simplification, which is beneficial in some cases and disadvantageous in others. The upper bundle of vertices in Figure 13 is such a case as they could be aligned better if the slope ladders were longer.

Figure 14 compares the outputs of Dyken et al. (b) with Kronenfeld et al. (c) and our harmonious algorithm (d) to the same input (shown in a) and simplified to about the same number of vertices. Quite clearly, the individual simplifications of individual isolines are valid in all three solutions. But only (d) successfully produces *families* of isolines. Consider the south-eastern slope, showing equally spaced input lines, which are moved away and towards each other in (b) and (c). Especially observe the glacial fan in the north-western portion

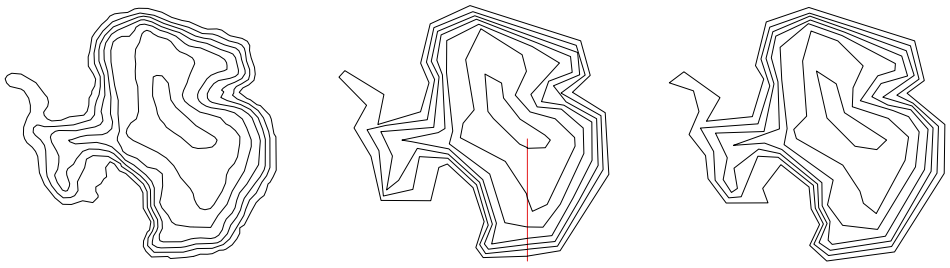
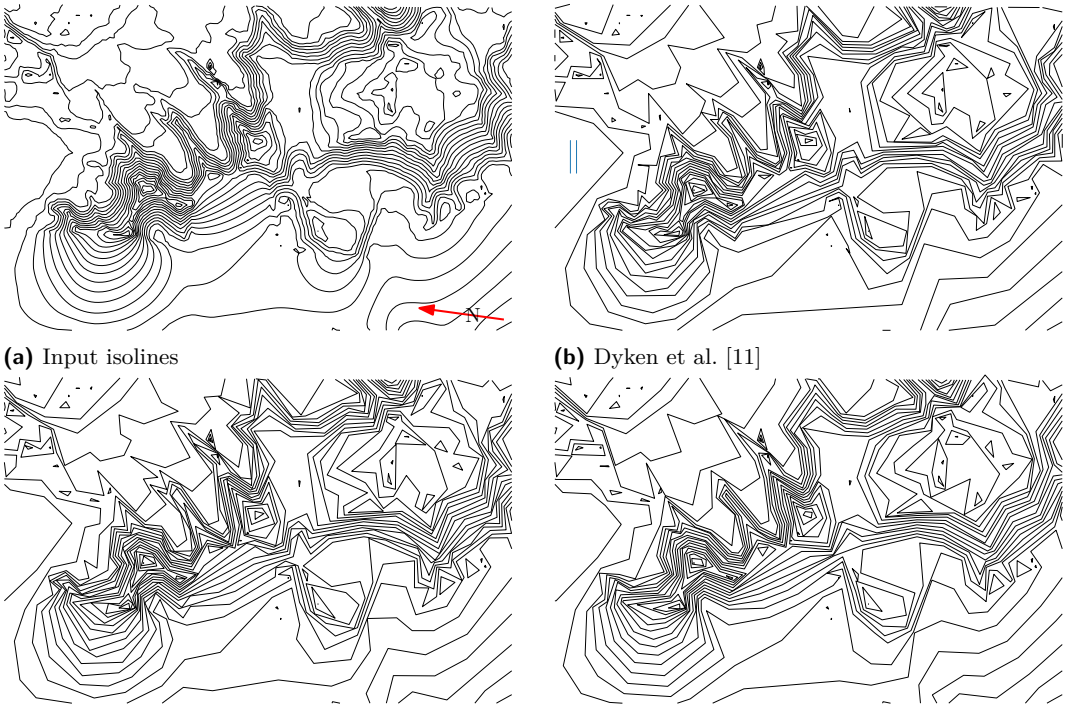


Figure 13 Left: input isolines consisting of 584 vertices. Middle: Van Goethem et al., 150 vertices. Right: our method with hybrid collapse, 146 vertices.



(a) Input isolines

(b) Dyken et al. [11]

(c) Kronenfeld et al. [25] (SymDiff1)

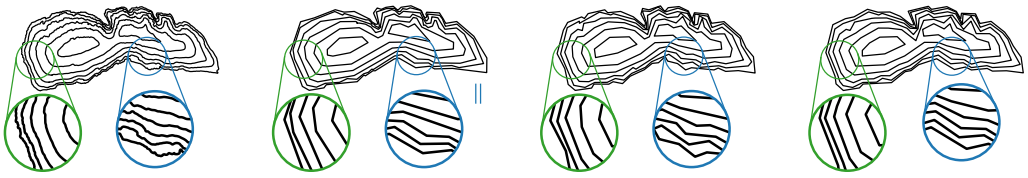
(d) Our harmonious algorithm with hybrid collapse

Figure 14 Dataset **MSY**. The input isolines have 7995 vertices. Figures (b) and (c) have $2^{10} = 1024$ vertices and (d) has 1023 vertices.

where it is easy to speak of an evolving family of curves for our approach (d) but not at all in (b) or (c). Also inspect the northern summit: the rough pentagonal shape of the summit can be ascertained in (a) and (d), but we see only rough irregularities in (b) and (c).

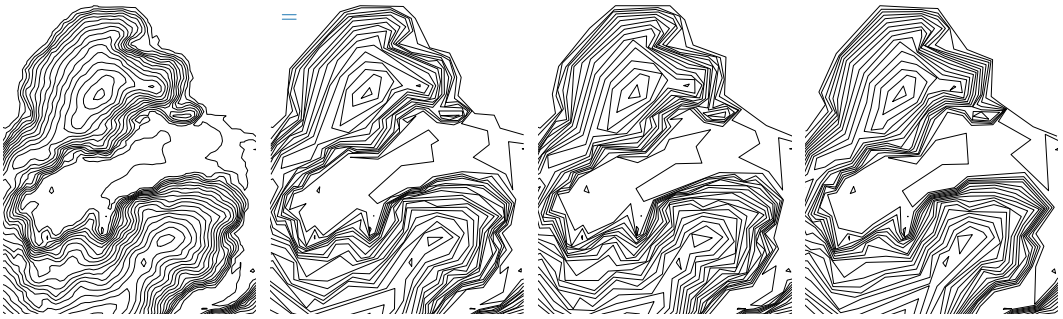
Figure 15 shows more scale-appropriate simplifications compared to the other figures. Here it is more difficult to see differences between algorithms; however, the results of our harmonious algorithm are still convincing. Compare the green highlight across the figures: in (d) the isolines are simplified to be parallel, but in (b) and (c) isolines bend at different places, adding noise without capturing input features. In the blue highlight we see that (d) aligns the isolines, but allows a deviation of the lowest isoline to capture a plateau in the input. The simplification in (c) captures both bottom plateaus, improving similarity with the input but reducing harmony, which matches the quantitative results of the previous section.

Figure 16 shows a close-up of Thurston Island, namely Hughes and Tinglof peninsulas



(a) Input isolines **(b)** Dyken et al. [11] **(c)** Kronenfeld et al. [25] **(d)** Harmonious

Figure 15 Selected isolines of dataset **SWI**. The input isolines have 20931 vertices and the simplifications have 200 vertices.



(a) Input isolines **(b)** Dyken et al. [11] **(c)** Kronenfeld et al. [25] **(d)** Harmonious

Figure 16 Extracts of Figure 17 and 18 (red rectangle).

with the flat area being the Henry inlet. As with the whole island it is ice-covered. The DEM-derived isolines of the input show that there are no sharp edges or rocks cropping out, so jarring discontinuities are not in the input and should not be in the generalized result. On first inspection it is quite obvious that Figure 16 (d) is the most ordered derivation from the input. Figures 16 (b) and (c) show zigging and zagging countour lines that do neither zig nor zag in unison. In detail, this leads to several deformations of the actual terrain: in (c), the northern slope of Hughes peninsula (the northernmost in the extract) shows a serrated curve edge, terraced and not aligned with each other, actually creating a little plateau that does not exist. A similar plateau-effect is created in (b) by an added vertex for each of the three lowest isolines on that slope, whereas the harmonised lines of (d) do not produce such artifacts. Also note how Dyken et al.'s approach works slightly more shape-preserving on some individual lines: the characteristic shape of the middle isoline of the Henry inlet is over-exaggerated in (c) and blunted in (d).

Finally, Figure 17 shows the input isolines for the area of Thurston island, including Peacock Sound and Sherman Island. The input is derived from a 10m resolution DEM for an area about 340km from east to west and 150km north to south (≈51000 km² is about the size of Costa Rica or Bosnia and Herzegovina). It is a test for how well the algorithm can work with realistic, detailed input data on a regional scale. In Figure 18, the detailed results of our two comparison methods and our own algorithm are provided. When zooming in and comparing details beyond the ones already highlighted in the excerpt, many areas are of about equal quality for all three solutions, such as the central ridge of the Walker Mountains. If we may direct the reader to the northern slope of Sherman island, the visual advantages of our method become apparent: the gentle undulations, which are kept in our approach, throw off Dyken et al., resulting in terrace-like structures and let Kronenfeld et al. show pointy rocky outcrops, illusions all of them. A further example of note is the south bank of the Morgan Inlet at the eastern part of Thurston island: the steep slope is well handled by the

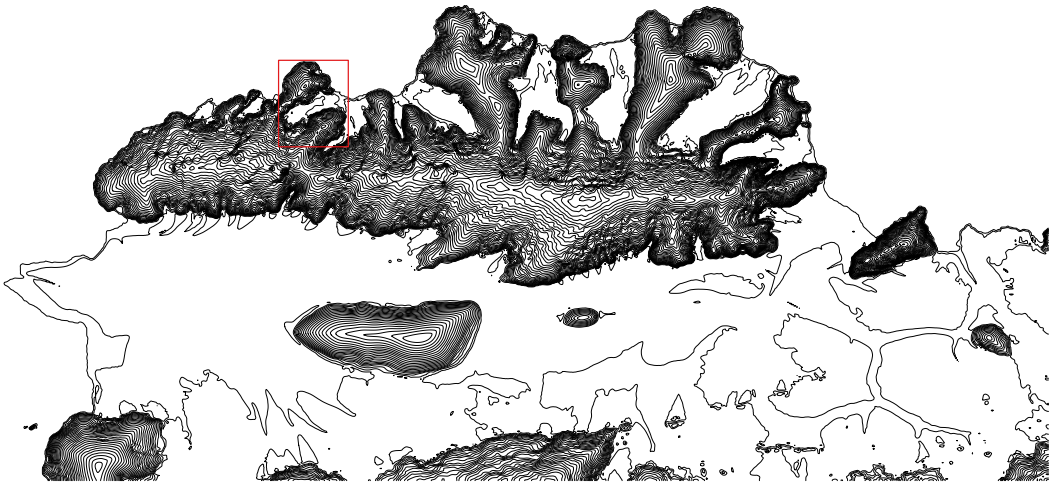


Figure 17 Dataset **LWI**. Input isolines of 139658 vertices.

our method, whereas both of the other candidates create readability problems even when zoomed in very closely.

4 Conclusion

In this paper we described a method for the simplification of isolines that takes their mutual relations into account and preserves their harmony. Our method is based upon the algorithm by Van Goethem et al. [15], but improves upon it in several important ways, thereby lifting it from an interesting proof-of-concept to a method fit for practical use on very large datasets. Our new simplification algorithm outperforms existing state-of-the-art methods on our measures of harmony, while compromising little on similarity. We can deliver topologically correct results on datasets with thousands of points within seconds and on a million of points in under half an hour. Furthermore, our method can handle sets of isolines with arbitrarily complex nesting structures.

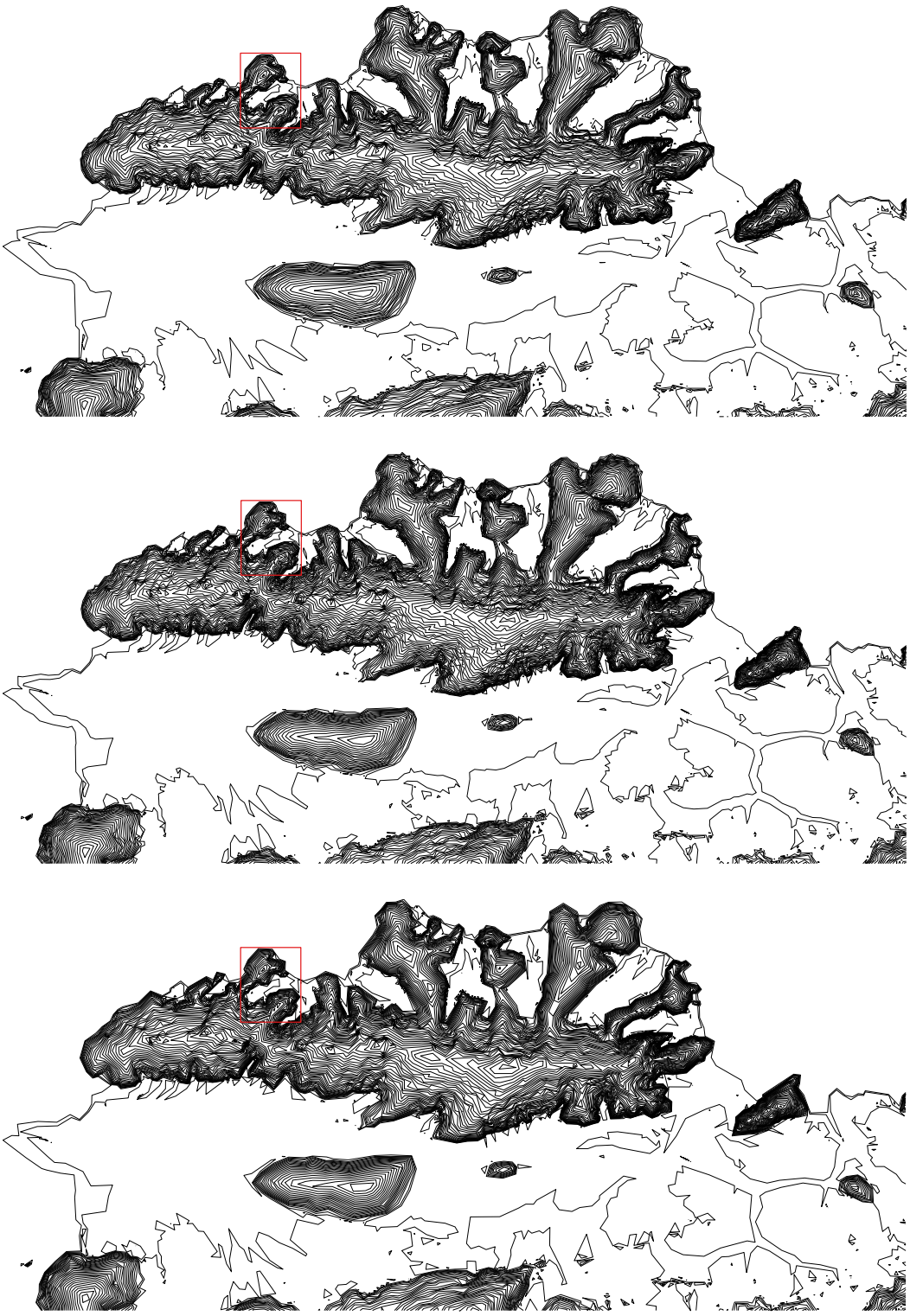


Figure 18 Simplifications of **LWI** (Figure 17). Top and middle have $2^{14} = 16384$ vertices and are computed by Dyken et al. [11] and Kronenfeld et al. [25] respectively. The bottom shows our harmonious simplification consisting of 16377 vertices using the hybrid collapse.

References

1 Sergey Bereg. An approximate morphing between polylines. *International Journal of Computational Geometry & Applications*, 15(2):193–208, 2005. doi:10.1142/S0218195905001658.

2 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational geometry: algorithms and applications*. Springer, third edition, 2008. doi:10.1007/978-3-540-77974-2.

3 Harry Blum. A transformation for extracting new descriptions of shape. *Models for the perception of speech and visual form*, pages 362–380, 1967.

4 Jean-Daniel Boissonnat, Olivier Devillers, René Schott, Monique Teillaud, and Mariette Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete & Computational Geometry*, 8:51–71, 1992. doi:10.1007/BF02293035.

5 Prosenjit Bose, Sergio Cabello, Otfried Cheong, Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, 4(4):554–566, 2006. doi:10.1016/J.JDA.2005.06.008.

6 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Bettina Speckmann. Locally correct Fréchet matchings. *Computational Geometry*, 76:1–18, 2019. doi:10.1016/J.COMGEO.2018.09.002.

7 Kevin Buchin, Yago Diez, Tom van Diggelen, and Wouter Meulemans. Efficient trajectory queries under the Fréchet distance (GIS cup). In *Proc. 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 101:1–101:4. ACM, 2017. doi:10.1145/3139958.3140064.

8 Kevin Buchin, Wouter Meulemans, André van Renssen, and Bettina Speckmann. Area-preserving simplification and schematization of polygonal subdivisions. *ACM Transactions on Spatial Algorithms and Systems*, 2(1):2:1–2:36, 2016. doi:10.1145/2818373.

9 Katrin Dobrindt and Mariette Yvinec. Remembering conflicts in history yields dynamic algorithms. In *Proc. 4th International Symposium on Algorithms and Computation*, volume 762, pages 21–30. Springer, 1993. doi:10.1007/3-540-57568-5_231.

10 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973. doi:10.3138/FM57-6770-U75U-7727.

11 Christopher Dyken, Morten Dæhlen, and Thomas Sevaldrud. Simultaneous curve simplification. *Journal of Geographical Systems*, 11(3):273–289, 2009. doi:10.1007/S10109-009-0078-8.

12 Andreas Fabri. 2D polyline simplification. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024. URL: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgPolylineSimplification2.

13 Efi Fogel, Ophir Setter, Ron Wein, Guy Zucker, Baruch Zukerman, and Dan Halperin. 2D regularized Boolean set-operations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024. URL: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgBooleanSetOperations2.

14 Arthur van Goethem, Wouter Meulemans, Andreas Reimer, and Bettina Speckmann. Simplification with parallelism. In *Proc. 23rd ICA Workshop on Generalisation and Multiple Representation*, 2020.

15 Arthur van Goethem, Wouter Meulemans, Andreas Reimer, and Bettina Speckmann. Harmonious simplification of isolines. In *Proc. 11th International Conference on Geographic Information Science*, pages 8:1–8:16, 2021. doi:10.4230/LIPICS.GISCIENCE.2021.II.8.

16 Alan H. Goldman. Aesthetic qualities and aesthetic value. *The Journal of Philosophy*, 87(1):23–37, 1990. doi:10.2307/2026797.

17 Eric Guilbert, Julien Gaffuri, and Bernhard Jenny. Terrain generalisation. In *Abstracting Geographic Information in a Data Rich World*, LNGC, pages 227–258. Springer, 2014. doi:10.1007/978-3-319-00203-3_8.

18 Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, DTIC, 1997.

19 Thijs van den Horst and Tim Ophelders. Faster Fréchet distance approximation through truncated smoothing, 2024. Preprint. arXiv:2401.14815.

20 Ian Howat, Claire Porter, Myoung-Jon Noh, Erik Husby, Samuel Khuvis, Evan Danish, Karen Tomko, Judith Gardiner, Adelaide Negrete, Bidhyananda Yadav, James Klassen, Cole Kelleher, Michael Cloutier, Jesse Bakker, Jeremy Enos, Galen Arnold, Greg Bauer, and Paul Morin. The Reference Elevation Model of Antarctica - Mosaics, Version 2, 2022. doi:10.7910/DVN/EBW8UC.

21 Ian M. Howat, Claire Porter, Benjamin E. Smith, Myoung-Jong Noh, and Paul Morin. The reference elevation model of Antarctica. *The Cryosphere*, 13(2):665–674, 2019. doi:10.5194/tc-13-665-2019.

22 Eduard Imhof. *Kartographische Geländedarstellung*. De Gruyter, 1965. doi:10.1515/9783110843583.

23 Menelaos Karavelas. 2D segment Delaunay graphs. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024. URL: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgSegmentDelaunayGraph2.

24 Menelaos I. Karavelas. A robust and efficient implementation for the segment Voronoi diagram. In *Proc. International symposium on Voronoi diagrams in science and engineering*, pages 51–62, 2004.

25 Barry J. Kronenfeld, Lawrence V. Stanislawski, Barbara P. Buttenfield, and Tyler Brockmeyer. Simplification of polylines by segment collapse: minimizing areal displacement while preserving area. *International Journal of Cartography*, 6(1):22–46, 2020. doi:10.1080/23729333.2019.1631535.

26 Thomas Mendel. Area-preserving subdivision simplification with topology constraints: Exactly and in practice. In *Proc. 20th Workshop on Algorithm Engineering and Experiments*, pages 117–128. SIAM, 2018. doi:10.1137/1.9781611975055.11.

27 Ravi Peters, Hugo Ledoux, and Martijn Meijers. A Voronoi-based approach to generating depth-contours for hydrographic charts. *Marine Geodesy*, 37(2):145–166, 2014. doi:10.1080/01490419.2014.902882.

28 Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. doi:10.1016/S0146-664X(72)80017-0.

29 Nicolas Regnauld and Robert B McMaster. A synoptic view of generalisation operators. In *Generalisation of geographic information*, pages 37–66. Elsevier, 2007. doi:10.1016/B978-008045374-3/50005-3.

30 Andreas Reimer. *Cartographic modelling for automated map generation*. PhD thesis, Technische Universiteit Eindhoven, 2015.

31 Xiaohua Tong, Yanmin Jin, Lingyun Li, and Tinghua Ai. Area-preservation simplification of polygonal boundaries by the use of the structured total least squares method with constraints. *Transactions in GIS*, 19(5):780–799, 2015. doi:10.1111/TGIS.12130.

32 Dražen Tutić and Miljenko Lapaine. Area preserving cartographic line generalizaton. *Kartografija i geoinformacije (Cartography and Geoinformation)*, 8(11):84–100, 2009.

33 Robert Weibel. Generalization of spatial data: Principles and selected algorithms. In *Algorithmic Foundations of Geographic Information Systems*, volume 1340 of *LNCS*, pages 99–152. Springer, 1996. doi:10.1007/3-540-63818-0_5.