Algorithms for Studying the Complexities of Isolines

Arthur van Goethem, Wouter Meulemans, Andreas Reimer, Bettina Speckmann
Eindhoven University of Technology
The Netherlands

ABSTRACT: *Isolines share link different places that share a common value. Efforts have been done to reduce the complexity of drawing lines during the measurement of geometric similarities. The isolines that have common scalar filed, the geometric similarity of the isolines and the separation is not easy. We should keep the similarities and structure intact which help to described the terrains. We have developed algorithm to address the complexity of isolines when studying the harmony. During the testing of the algorithms we found it more efficient.*

Keywords: Simplification, Isolines, Harmony

Received: 24 June 2021, Revised 11 August 2021, Accepted 28 August 2021

DOI: 10.6025/jmpt/2021/12/4/110-124

1. Introduction

Isolines are among the most common and effective cartographic techniques for visualizing scalar field-type data, such as terrain, precipitation, air pressure or, more abstractly, travel times. Isolines as a visualization method share and mirror an aspect of fields themselves: only the multitude of isolines signifies the phenomenon and the objects that it comprises. A single spot height is insufficient to understand the lay of the land, and equally insufficient is the individual isoline. It follows naturally, that the visual and the geometric relationships between individual, pairs, and groups of isolines are essential to their cartographic effectiveness.

Geographic data need to be represented at different scales for specific tasks. The processes that transform the visualization of geographic datasets due to scale-changes are subsumed under the term generalization. Our approach aims to partially capture interdependencies in the form of "harmony" of isolines while generalizing them. Visual harmony as a concept alludes to neighboring isolines behaving in a similar manner where such behavior is warranted. In cartographic practice, questions of harmony used to be addressed by the human cartographer. Nowadays isolines are usually generated automatically by interpolation processes from scalar fields, with the most common example being Digital Terrain Models (DTM) [32]. As most progress in automated generalization has been made for topographic maps, research in automatization of isolines so far has concentrated on the generalization of terrains, with systems of contour lines being one of the major outputs [16].

There are two main advantages to generalizing isolines instead of resampling a scalarfield (usually a DTM) and then interpolate fresh isolines: (1) the ability to *directly* keep relations intact within a multi-resolution database, and (2) the avoidance of visual jumps and resampling artifacts. Consider, for example, a river which must always run through the deepest part of a valley and as such must be tied to the isolines. When resampling a DTM and generalizing the drainage polylines independently, the generalized rivers might run over the slopes of the freshly derived contour line system [25]. Hence, until a few years ago, many national mapping agencies either kept their manually generalized isolines for their existing scale ranges and did not automatically derive them for production of topographic maps, or employed sophisticated techniques to tie freshly generated contour lines to pertinent database objects and/or the DTM (such as the GAEL-model for Object-Field Relation Preservation [9]). Such conflation methods are not applicable to most other scalar fields due to lack of ancillary data. They have also been out of reach for most GIS users, although some open-source plugins for DTM-based tasks have been released recently [25, 30].

Results and organization. In this paper we propose an algorithm to simplify a whole system of vector isolines while considering harmony explicitly. In Section 1.1 below, we present a framework which allows us to consider harmony as an element of cartographic merit of systems of contour lines. We describe our harmonious simplification algorithm in Section 2 and present experimental results in Section 3. Our examples focus on contour lines, but we note that our algorithm works with any isoline vector input.

Our method allows us to apply and refine many heretofore neglected design rules (such as those related to harmony) for isoline systems. It also naturally supports animated morphs between scales without loss of connections. Last but not least, since our algorithm is targeted towards vector data, it allows us to apply locally adaptive generalization.

1.1 Harmony

While the word harmony has a wide range of meanings, we are concerned only with harmony in relation to isolines. On first sight, the harmonious visual interaction between isolines looks like a lofty or imprecise concept. However, the notion that a pleasing and informative repetition that goes beyond mere parallelism is a crucial design goal, permeates the literature. In Swiss cartographic debates of the early 20th century, harmony was explicitly named as an intermediary design goal, which directly influenced academic and production mountain and terrain cartography [27]. To move from the qualitative and aesthetic concept of harmony to actionable requirements, we derive the pertinent design rules from cartographic examples and cartographic literature. From those design rules, we move on to measurable qualities.

The basic idea of our approach of conceptualizing harmony is a modification of the process used by Reimer [24] for deriving and modeling harmony for chorematic diagrams. Reimer follows Goldman [13] and observes that the general merit of a map can be operationalized by linking objective, measurable qualities to aesthetic qualities of varying abstractness. This is done hierarchically: the general merit of a map is a function of its *accuracy*, *expressiveness*, and *beauty*, with *originality* an optional addition that may well be zero, and is indeed zero in our case. The first three are the first tier aesthetic qualities. To each of them, Reimer [24] links a number of second tier aesthetic qualities: more precise, but still not directly measurable. The third hierarchical tier then links measurable qualities to the second tier. While the first and second tier are capturing evaluative concepts which apply to every cartographic expression, the third tier is specific to a type of map or map element. That is, the measurable qualities as formalized in the third tier apply to a specific map type, which are isolines in our case (see Figure 1). Once we have formalized the third tier for isolines, we can define harmony as a combination of third tier elements.

Tier 3: measurable qualities for isolines. For the case of contour lines, the accepted master both on aesthetic as well as scientific design rules is Eduard Imhof [18]. The three basic overarching dimensions of Imhof's conceptualization of a good contour line system arguably match directly to Goldman's top tier: "Auswertbarkeit" corresponds to accuracy, "Lesbarkeit" corresponds to expressiveness, and "Empfinden" corresponds to beauty. Imhof operationalizes these dimensions into concrete global qualities, namely:

G1 Scale-appropriate terrain data.

G2 Contour interval matching the target scale and depicted terrain.

a Round numbers for easy counting preferred.

b Area between isolines shall be neither too big nor too small.

c Harmony between all elements of the system.

G3 Easy readability, including generalization, where needed.

As we are interested in generalizing isolines and assuming a stable **G1** and **G2** setting, we are especially interested in his comments on **G2c** and **G3**. Below we summarize Imhof's design rules for small-scale contour line generalization [18]:

R1 For each small area of terrain, design the whole line system together.

R2 Smooth over small jagged parts in a single line if it is not reflected in neighboring lines.

R3 Neighboring contours should never touch. Exceptions for alpine regions (cliffs, karren).

R4 Shapes should reflect reality; sometimes with pointy and jagged parts, sometimes gentle curves, but neither to the exclusion of the other.

R5 Use ancillary data like terrain crests or breaks of slope when drawing the contour lines.

R6 If a small form cuts across several contours, it should be treated as a whole. Either retain and draw it everywhere or remove it.

R7.1 Retain prominent, convex, terrain features which are much more visible in-situ than hollow forms.

R7.2 Emphasize important hollow forms, widening them for legibility if necessary.

R8 Minimize geometry changes when following the other rules.

For **R3** Imhof specifically highlights that the whole line system must be brought under control and carefully adjusted, not just the overlapping pairs. This is a recurring theme with Imhof's design rules which also appears in label placement. We populated Tier 3 with the measurable properties discussed by Imhof, either in the form of rules or in the form of examples. We added statistical verity as well as marginal information.

Imhof does not necessarily provide evaluation functions or even parts thereof together with his rules. An example would be **G3b**, namely the ratio of area to line, which must be somehow balanced, but no further information is provided. In Figure 1 we highlighted those Tier 3 properties in blue for which Imhof gives sufficient guidance. We further highlighted those properties in green for which sufficient guidance can be found in the cartographic literature. There remain three properties for which Imhof presents rules, but not sufficient guidance, namely curve adaption, parallelism, and area to line ratio.

We are now ready to define *harmony of isolines*: a melange of elements of poignancy and shapeliness. Following the hierarchy, harmony is potentially measurable by evaluation functions for generalization solutions, curve adoption and parallelism.

1.2 Related work

A very good overview until 2014 of automated terrain generalization, of which contour line generation is a subset, can be found in [16]. Since then, recent work was done, for example, on isobath generalization [15, 28], continuous generalization via morphing [10], improving conflation of raster DTMs with contour lines [25, 30] and generation of supplementary contour lines [26]. These examples highlight a growing interest into automated sea-chart production, online mapping and continued support for NMA-demand via ancillary data-driven processes based on DTMs for multiple resolution databases. Isobaths are often considered a special case, due to the peculiar needs for safety constraints [15] as well as being not entirely treated as representations of a continuous scalar field but rather discrete maritime objects. Although direct generalization of isobaths from other isobaths in the vector realm is more common than in current dry-land cartography, they are thus not too much concerned with how the individual line shapes interact. Current morphing approaches such as [10] use a vertex matching between origin and target scale and then interpolate on that matching, with few regards to cartographic design rules.

Moreover, most proposed techniques for the simplification of isolines (e.g., [10, 20]) focus on individual isolines without explicitly considering the relationship between adjacent isolines. Recently, van Goethem et al. [11] proposed a technique for simultaneous simplification of isolines with a focus on parallelism. They strengthen the paralellism between adjacent isolines while simplifying the input. However, the relative positions of the vertices on adjacent isolines is not explicitly taken into account, thereby addressing but one facet of harmony.

In recent years several automated simplification approaches have been proposed that allow the introduction of new vertex locations to more closely approximate the original input at lower complexity simplifications. The potential new vertex positions are usually limited, for example using criteria such as maintaining local topology [7, 12], interpolation between scales [10], or using an underlying grid [23]. An alternative criterion that is often used is area preservation (e.g., [6, 12, 19, 22, 29, 31]). Area preservation mandates that the area covered by the simplified feature is exactly equal to the area covered by the original feature. Area preservation ensures that features cannot arbitrarily increase or decrease in size during the simplification process and can be used to significantly reduce the search space for new vertex positions. These criteria are often combined together with a standard step to introduce new vertices. One of the more prominent techniques used for this is the *edge collapse* (e.g, [19, 31]). In an edge collapse a sequence of three consecutive edges is replaced by two new edges, implicitly "collapsing" the middle edge to a vertex. Our proposed technique follows these principles of area-preserving edge collapses, but performs such operations in parallel, so as to promote harmony.

Computing an optimal simplification under the criteria of area preservation is a hard computational problem. Bose et al. [4] showed that even when solely considering an *x*monotone path computing a simplification of *k* links that is area preserving is NP-hard.

Löffler and Meulemans [21] showed that even when the resulting area-preserving simplification is restricted to only having (horizontal and vertical) edges on an orthogonal grid the problem is NP-hard. Similar results appear when considering simplification in a network. Estkowski and Mitchell [8] have shown that minimizing the number of (degree-2) vertices in a network subject to a given error bound while preventing intersections is NP-hard to approximate to within a factor $n^{\frac{1}{5}-\delta}$, $\delta > 0$. The results discussed above underpin the necessity for efficacious heuristics, such as our proposed method, for computing (harmonious) simplifications.

2. Harmonious Simplification Algorithm

We present a simple, efficient algorithm to compute the harmonious simplification of a sequence of polygonal lines. We assume the polylines represent a consistent part of the terrain (a common snippet of isolines). We note that our techniques readily apply to polygons as well, and can be generalized to more complex arrangements of isolines. The exact steps of such generalization are out of scope for this paper and will be addressed in future work.

Our algorithm consists of two main phases: (1) preprocess the polylines to identify so-called *slope ladders*; (2) repeatedly *collapse* slope ladders to reduce complexity while maintaining harmony. Our initial input is a sequence of polylines $P = <P_1, \ldots, P_k>$ where each $P_h = <v_{h,1}, \ldots v_{h,nh}>$ is a polygonal line defined by n_h vertices in the plane. Each polyline represents an isoline and they are ordered by the height they represent, thus consecutive isolines are also consecutive in the order. We assume that all polylines are given in a consistent orientation; it is straight-forward to ensure this assumption is always met. Because the polylines represent isolines, they do not intersect each other nor self-intersect.

2.1. Preprocessing: deriving slope ladders

To maintain harmony during simplification we first need to detect the harmony that is present in the input. We determine a *matching* between the vertices of every consecutive pair of isolines and then combine these into *slope ladders* stretching multiple isolines.

Matching. The matching between the isolines describes how the adjacent isolines are related to each other, particularly which pairs of vertices are considered "nearby", and thus implicitly describes the terrain represented by the isolines. Such a matching could also be derived from slope information if available. However, lacking such information, several algorithms exist that can compute such a matching based on the geometry of the isolines alone. In this paper we focus on such geometric algorithms, and particularly consider Dynamic Time Warping (DTW) [3] and the (discrete) Fréchet distance [2]. In this paper, we use Fréchet distance to refer to the discrete variant, rather than the continuous variant.

We require that a matching between polylines P_h and P_{h+1} maps every vertex of P_h to at least one vertex of P_{h+1} and each vertex of P_{h+1} to at least one vertex of P_h such that the matching is *monotone*. That is, if vertex $v_{h,1}$ is matched to $v_{h+1,j}$ then every vertex

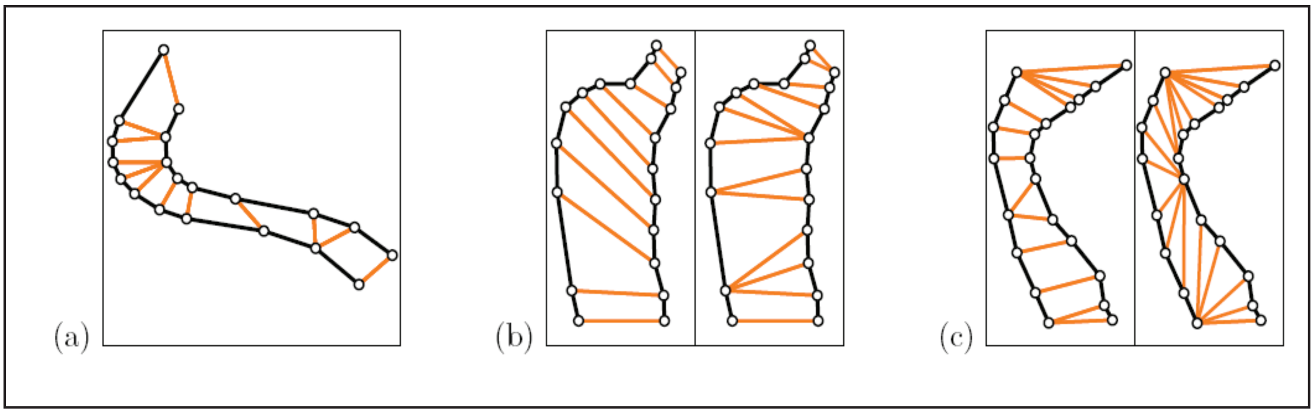


Figure 2. (a) The polyline matching should be monotone. (b) The DTW matching (left) minimizes the number of matched pairs instead of preserving locality (right). (c) Matchings with the same Fréchet distance; the left matching is locally correct

$v_{h,i'}$, with $i' > i$, can only be matched to vertices $v_{h+1,j'}$ where $j' \geq j$, and similarly every vertex $v_{h+1,j''}$, with $j'' > j$ must be matched to vertices $v_{h,i''}$, where $i'' \geq i$. Intuitively this gives rise to a sequence of matchings that have a consistent order along both polylines (see Figure 2(a)). Both DTW and the Fréchet distance adhere to these requirements.

As we assume the terrain to be locally consistent with the isolines, matched vertices should generally be nearby. While both the DTW and the Fréchet distance capture this locality to some degree, both are not completely satisfactory. Particularly the DTW minimizes the *sum* of distances between all matched vertices. Therefore the DTW tends to prefer fewer pairs even if the pairs of vertices are not close together (see Figure 2(b)). In contrast the Fréchet distance focuses purely on the *maximum* distance between any pair of matched vertices (the *bottleneck*). However, no further distinction is made between all matching that minimize the bottleneck (see Figure 2(c)). This may lead to matchings that may not necessarily describe similarity well [5]. To overcome this issue, Buchin et al. introduced locally correct Fréchet matchings (LCFM) [5]. These matchings recursively enforce the bottleneck, ensuring the lowest distances between matched pairs are achieved for all pairs. Such a locally correct (discrete) Fréchet matching can be computed in the same time as the Fréchet distance.

We observe, however, that using the Euclidean distance for the matching may still lead to counterintuitive results as the matching line is exterior to the area bounded by the two isolines (see Figure 3). To prevent such counterintuitive matchings, we use the geodesic distance in the bounded area instead. We abbreviate the matchings as ELCFM and GLCFM to distinguish the Euclidean and Geodesic variants, and again use only the discrete variants.

Slope ladders. We combine the pairwise matchings between consecutive polylines to obtain a *slope skeleton* of the hillside described by the isolines. The slope skeleton relates the underlying geometry across all isolines (**R1**). For each edge of a

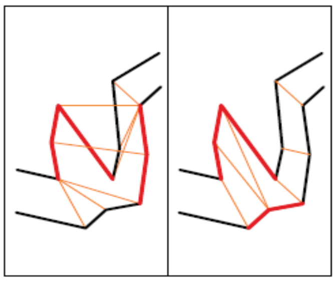


Figure 3. Locally correct Fréchet matchings using the Euclidean distance (left) and using the geodesic distance (right)

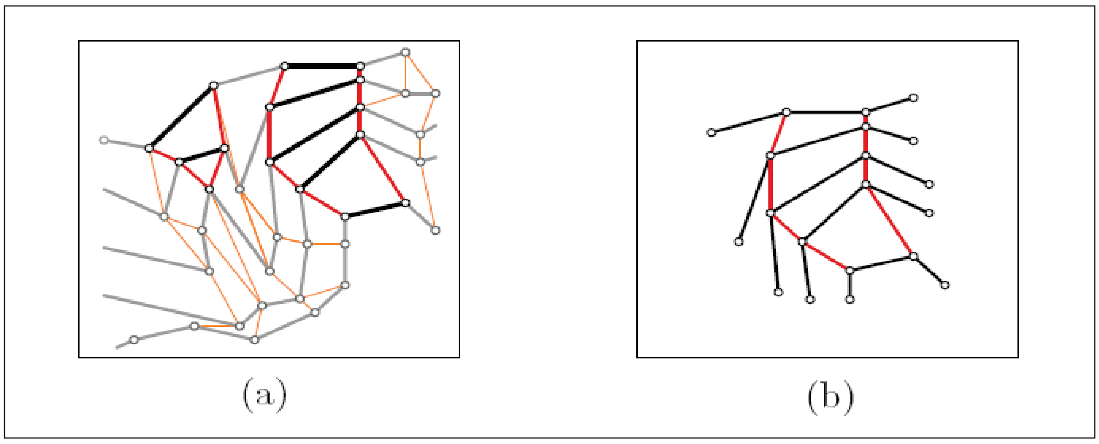


Figure 4. (a) Two examples of slope ladders in this skeleton, one of complexity 2 and one of complexity 5. (b) The rungs of the slope ladder (purple) and the associated sequences of four vertices

polyline we distinguish two cases. Either both endpoints of the edge are mapped to the same vertex on the adjacent polyline (creating a triangle) or they map to two consecutive vertices (creating a quadrilateral).

We maximally join all quadrilaterals and triangles that share a polyline edge into what we call a *slope ladder*: a sequence of quadrilaterals, possibly capped with triangles (see Figure 4(a)). A slope ladder can intuitively be thought of as describing a sloped face of the hillside. The polyline edges in a slope ladder we refer to as the *rungs* of the ladder (see Figure 4(b)). Each rung is associated with four vertices <*s, t, u, v*>: the endpoints of the previous edge, the rung, and the next edge of the polyline. The complexity of a slope ladder is its number of rungs.

The slope skeleton naturally, uniquely, and fully decomposes into slope ladders: every polyline edge is a rung in precisely one slope ladder. We can straightforwardly compute the slope ladders by starting a new slope ladder from an edge which is not assigned to a slope ladder yet, and using the matching with adjacent polylines to find new edges of the ladder, until the matching matches both endpoints to the same vertex.

2.2. Iterative Reduction: A Single Collapse

We simplify the polylines while ensuring that matched vertices are either maintained or removed as a whole (**R6**). This explicitly enforces harmony as we simplify the isolines. We do so by iteratively *collapsing* slope ladders. To further strengthen harmony we place new vertices such that they form a more visually striking cross-isoline feature. Particularly we ensure that newly placed vertices are collinear.

Area-Preserving Collapses. To collapse a slope ladder, we replace each of its rungs by a single vertex. Consider a rung *tu* with associated vertices <*s, t, u, v*> of polyline P_h. A collapse removes *t* and *u* from P_h and inserts in their place a single new vertex *p* (see Fig. 5(a, c)). The new vertex may be placed freely and need not be restricted to the input vertices. In line with other algorithms, we require area preservation for each isoline, such that the area between two isolines remains the same. Let *A* be the signed area of the region enclosed by the segment *sv* and the segments between vertices *s* to *v* in reverse order (see Figure 5(a)). This is the area that would be lost (respectively gained) between the adjacent isolines. We place the new vertex *p* such that an equivalent area is gained (respectively lost). As *spv* forms a triangle and *sv* is a fixed base, the height of the triangle with respect to this base is $h = 2A/|sv|$. Thus the placement of *p* is limited to a line parallel to the segment *sv* at height *h* (see Figure 5(b)). We refer to it as the *area-equivalence line* for rung *tu*: any *p* on this line ensures exact area preservation (see Figure 5(c)).

Harmonious candidate positions. To promote harmony we simultaneously select the position of the new vertices for all rungs (**R1, R6**). If there is only one rung in the ladder then we simply use the area-equivalence line and sample it to find the optimal location. In the following we assume there are at least two rungs. We have two criteria for the placement of the new vertices: (1) the position should reflect the underlying geometry well; (2) the positions of the new vertices should be harmonious. We enforce the second criterion by requiring that all new points are collinear. Given this constraint we then optimize

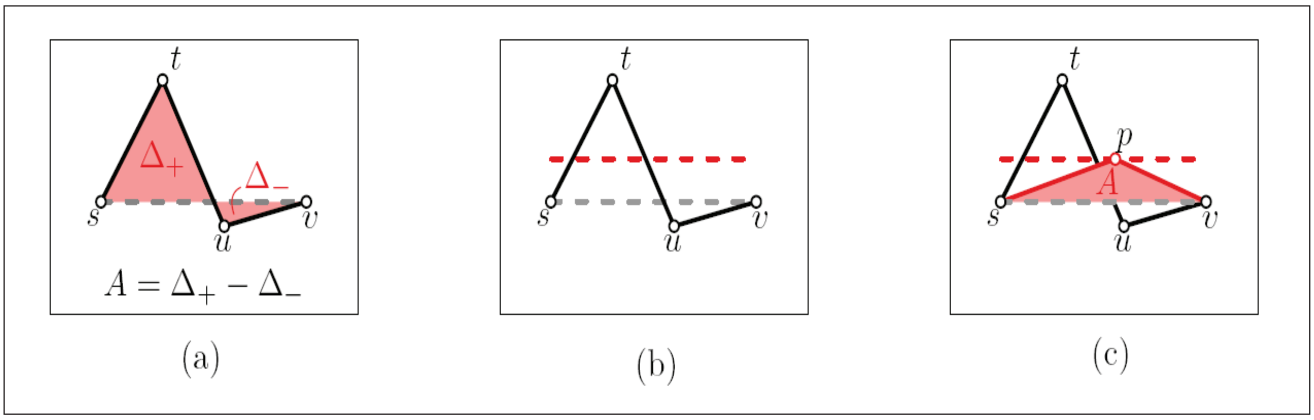


Figure 5. (a) Area contributed by the original rung and its adjacent edges. (b) The area equivalence line. (c) Any position of the area-equivalence line creates a triangle that contributes the same area as before. (Suboptimal position selected for visualization purposes)

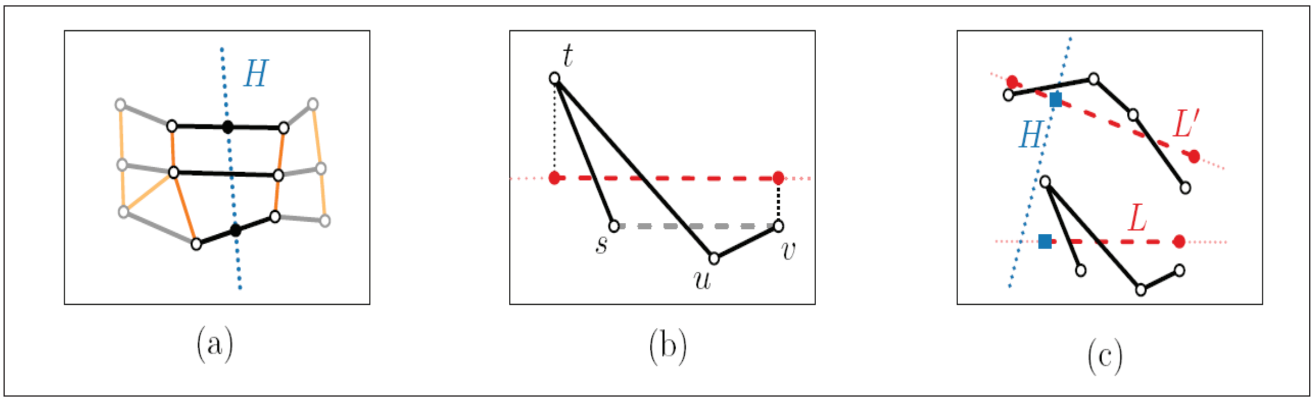


Figure 6. (a) *Harmony line H* for a slope ladder. (b) *Span* (dark red dashed) of a single rung on the area-equivalence line. (c) The new position (blue boxes) is the intersection of *H* and *L* (/*L*2) snapped to the nearest point on the *span*. (Arbitrary harmony line for visualization purposes)

for the first criterion (**R4, R8**).

As we require the vertices to be collinear, the only freedom left is to fix the line on which the new vertices will be placed. This line should ideally be perpendicular to the isolines themselves, to ensure good alignment of the incident edges for the new vertices. To this end we construct an initial *harmony line H* through the midpoints of the first and last rung in the slope ladder (see Fig. 6(a)). We assume the initial harmony line *H* intersects the area-equivalence line of each rung; if it does not, we do not allow collapsing this slope ladder.

The placement of the harmony line directly defines the placement of all vertices. Consider a single rung. Let è'*s, t, u, v*é' be the consecutive four vertices along the associated isoline that are the endpoints of the rung, the previous edge and next edge. Let *L* be the area-equivalence line belonging to the rung. The *span* of rung *tu* with respect to *L* is the maximum interval on *L* spanned by a pair of vertices from è'*s, t, u, v*é' (see Fig. 6(b)). If the intersection of *H* with *L* lies inside the span of the rung, then the position *p* at which the new vertex will be placed is the intersection of *H* and *L*, otherwise *p* is the closest endpoint of the span to the intersection of *H* and *L* (see Figure 6(c)).

Directly using the initial harmony line is too restrictive. To improve flexibility we allow the harmony line to be offset (perpendicular to its orientation) to find an optimal placement of the new vertices. For each offset of *H* we compute the new vertex locations and how well they approximate the original isolines. As a measure of the induced error per rung we use the directed Hausdorff distance from the newly inserted two edges to the represented section of the original polyline . That is, we measure the longest distance from a point along the new geometry to the original polyline. This is equivalent to computing the minimum epsilon-band around the represented section that contains the new edges. The total error induced by an offset of the harmony line is the maximum error over all rungs.

We sample offsets at regular distances, limiting the maximal offset in either direction to the extremal offset where at least one new vertex lies inside the span of the respective rung. We select the offset of the harmony line that minimizes the induced error.

Maintaining a Matching. To compute the directed Hausdorff distance to the original polyline, we need a matching of the current geometry and the original polyline. Initially, this matching is the identity. We update the matching iteratively during the algorithm per rung of the collapsed ladder. Let $<s, t, u, v>$ denote the four vertices associated with rung u; let μ denote the current matching. Moreover, let max $\mu(w)$ be the last vertex on the matched geometry that is matched to w. Similarly min $\mu(w)$ is the first vertex matched to w. As tu is replaced by a new point p, the main concern is to re-assign the vertices in $\mu(t)$ and $\mu(u)$ to s, p, or v such that the entire matching remains monotone. We compute the ELCFM between the new geometry $<s, p, v>$ and the original polyline from max $\mu(s)$ to min $\mu(v)$. This "local" ELCFM is used to update the matching μ. We use the Euclidean variant as the geometries may intersect and hence geodesic distances are unsuitable.

2.3. Iterative Reduction: Overall Algorithm

Our algorithm iteratively collapses slope ladders as detailed above. We now describe the remaining high-level algorithmic details: selecting the slope ladder to collapse, criteria for stopping, and how to efficiently avoid intersections.

Selecting a Collapse. We greedily select the collapse that causes the least visual change as measured by the (average) symmetric difference (or areal displacement) between the current geometry and the geometry after collapse; this is similar to existing iterative approaches, e.g, [6, 19]. As ladders with higher complexity naturally incur more (total) symmetric difference but also reduce the complexity (number of vertices/edges) more, we divide the total symmetric difference by the number of rungs c, to obtain the average symmetric difference per rung.

Note that our method for selecting the collapse geometry ensures that the result remains close to the input geometry, while our selection strategy ensures little visual change with respect to the current geometry. The advantage of the latter as a high-level strategy is that the sequence of simplifications changes as little as possible from step to step, thus improving the "stability" of the results. Simplifications of almost the same complexity will have similar layouts. Notably this is not guaranteed by the Hausdorff distance in the collapse computation as it compares only to the original polylines.

Stopping criteria. There are various options that may be used as a stopping criterion for the algorithm. We could (1) perform operations until the collapse cost exceeds a certain threshold, (2) disallow collapses that cause too large a Hausdorff distance, or (3) stop when the total number of vertices is at most a given parameter $z \geq 2k$. Following the techniques of [7], we observe that we can easily keep track of the performed operations in a data structure to efficiently reconstruct all intermediate results.

Avoiding intersections. To avoid introducing intersections between isolines or within the same isoline (**R8**), we perform only collapses that do not create intersections. We follow the algorithm described by [6] and maintain "blocking numbers": this number stores the number of edges that are intersected by the new geometry if the collapse was performed. Thus, we perform only collapses with blocking number zero. The main idea is that, after initializing these, collapses change only few edges and only these changing edges need to be considered to update the blocking numbers. This is more efficient than testing intersections from scratch.

Running-time analysis. We assume that there are k polylines of n_h vertices each, with $n = \sum_{h=1}^{k} n_h$ the total number of vertices. Our algorithm runs in $O(n^2 \log n)$ time, using $O(n)$ space. We first compute slope ladders with the GLCFM in $O(n^2 \log n)$ time using the data structure of Guibas and Hershberger [14] and the algorithm by Buchin et al. [5].

We then initialize all collapses in $O(n^2)$ using the approach of Alt et al. [1] to compute the Hausdorff distance. Finally, we iteratively collapse the best slope ladder and update any affected ladders in total $O(n^2 \log n)$ time.

3. Results and Discussion

We implemented our algorithm1 and tested[1] it with various isolines. We first qualitatively compare the results on different sets of isolines through a visual exploration of the properties.

We then quantitatively measure how well harmony is maintained using our method of collapsing slope ladders compared to simplifying each isoline separately.

We showcase and measure results for eight different inputs. The first are two digitized textbook examples, namely two sets of isolines used by Imhof to illustrate isoline generalization practices [18, page 155]; we denote these by **IMS** (slope) and **IMH** (hilltop). **AH1** is a digitized set of contour lines for the whole of Antarctica at 500m contour intervals. The other five are snippets from the DEM of Antarctica [17], deriving isolines at a 150m contour interval using QGIS v.3.16; we denote these snippets by **AS1** through **AS3** (slopes) and **AH2** through **AH3** (hilltops). All inputs are illustrated in Figure 7. The algorithm discussed considers polygonal lines, hence for our experiments we open any closed isoline. Though out of scope, the concepts can readily be extended to closed isolines and in future work this restriction will be lifted.

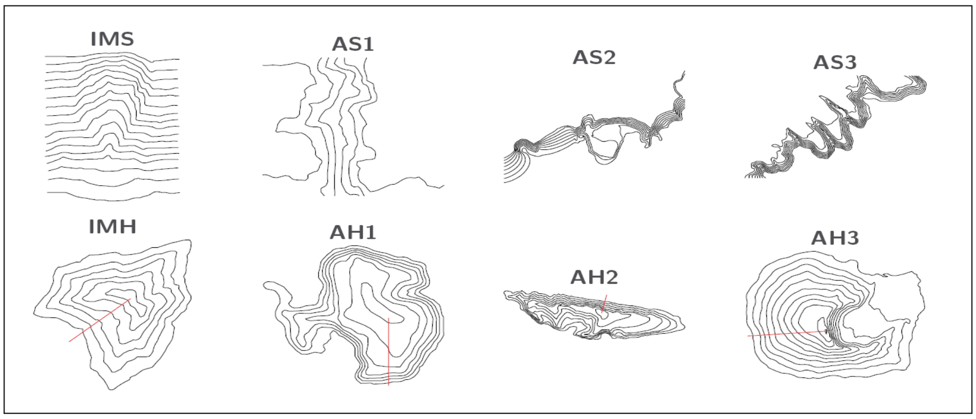


Figure 7. Inputs used in this paper: four slopes (top) and four hilltops (bottom). Complexities range from 442 (AS1) to 2071 (AS3). Red lines indicate where hilltops were opened into polylines

3.1. Qualitative Discussion

Examples by Imhof. We start our visual investigation with **IMH**; see Figure 8. We directly observe that the effect of harmonization is minor when the detail-level is high. Due to the large number of vertices in the simplification, the harmony is readily

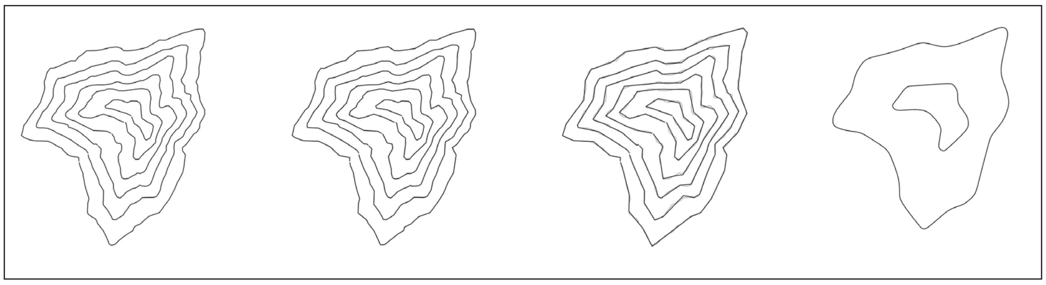


Figure 8. (left) IMH with 664 vertices. (middle left) Simplification at 348 vertices does not visually benefit much from harmony. (middle right) Simplification at 150 vertices, with harmony emphasizing the underlying patterns. (right) Example simplification as given by Imhof (from [18])

[1] https://github.com/tue-alga/harmonious-simplification (v0.1)

present as there are sufficient vertices to always have nearby vertices on the adjacent isoline. Isoline detail is the primary factor. However, when we reduce the complexity further, the underlying patterns in the isolines become emphasized through harmony. As vertices are placed at similar locations whenever isolines trace similar boundaries, higher-level structures visually start to form a coherent whole strengthening the relationship between adjacent isolines. The effect is visible for both **IMH** and **IMS** (see also Figure 13) giving credit to the suitability of the approach.

Visual complexity. When considering more tight-packed slope lines, the benefits of harmony are even more apparent. In Figure 9 we compare two simplifications of **AS3** with approximately the same number of vertices. One of the simplifications is computed with our algorithm, the other uses the same simplification steps but collapses edges independently during the process. We observe that enforcing harmony captures the strong similarity between the isolines. Consequently, similar sections are also highly similar in the simplification in contrast to the independently simplified isolines. This coherence (1) strongly reinforces the underlying patterns, (2) leads to a lower visual complexity as isolines can implicitly be grouped, and (3) emphasizes the object represented by the group of isolines in contrast to the isolines themselves. The lack of harmony in the alternative result hinders the visual perception of the isolines as representing a single object as the visual clutter hides underlying patterns.

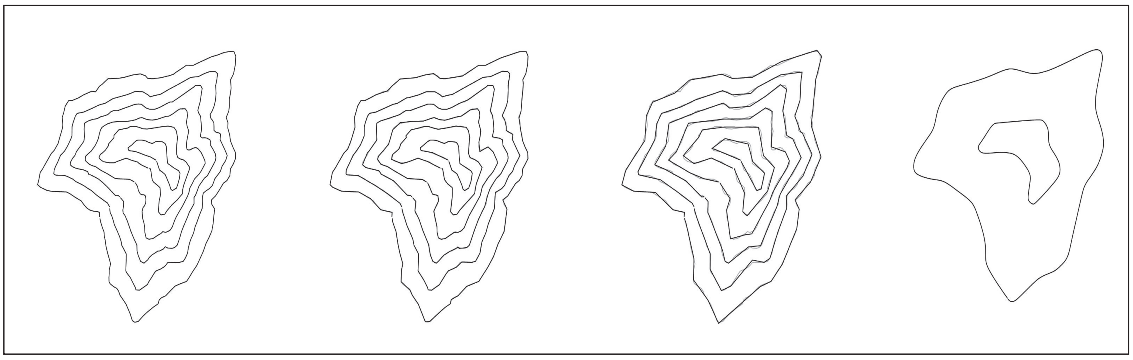


Figure 9. (left) AS3 with 2071 vertices. (middle) Simplification with 246. (right) Simplification with 247 edges using our approach but collapsing a single edge at a time

Limitations. Our algorithm sometimes makes minor mistakes that are undesirable (Figure 10). Particularly, the individual simplification may be suboptimal compared to a potential individual simplification as the error margin could trivially be lowered by shifting single vertices (Fig. 10(a/c)), or insignificant edges are maintained even though they do not contribute to the overall simplification (Fig. 10(b)). We believe these may most likely be traced back to the following issues. By strictly enforcing area-preservation it is possible that a vertex is placed to compensate for a region being cut off through simplification (a). The problem can be exaggerated due to the stringent requirement that all newly placed vertices need to be on a single

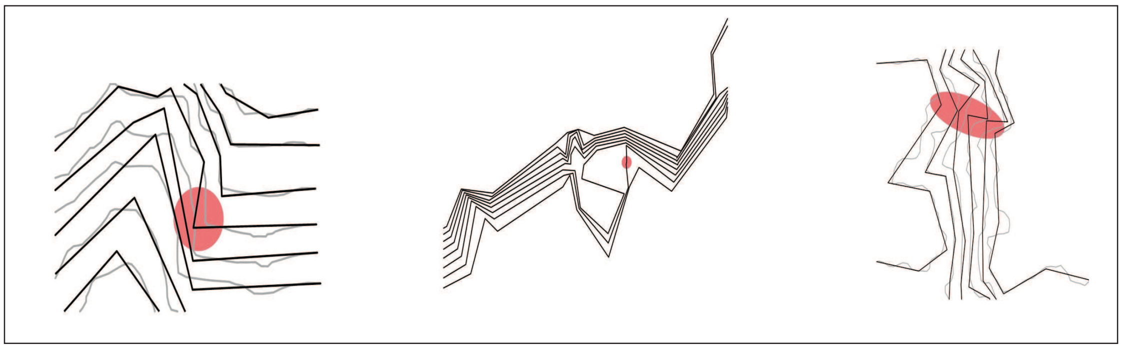


Figure 10. (left) area-preservation requirements may lead to poor vertex placement (IMS - 98 vertices), (middle) minor edges despite a high level of simplification (AS2 - 123 vertices), (right) forcing vertices to be one a single-line can cause unnecessary errors (AS1 - 50 vertices)

line (c). This may also affect the (im)possibility to simplify certain sections in union without a high error margin for one of the isolines, potentially leading to unnecessary minor edges (b).

In future work, we plan to investigate more flexible treatment of slope ladders to overcome these issues. Overall we believe the method shows great potential as witnessed by its ability to suitably simplify many different isoline structures (see Figure 11).

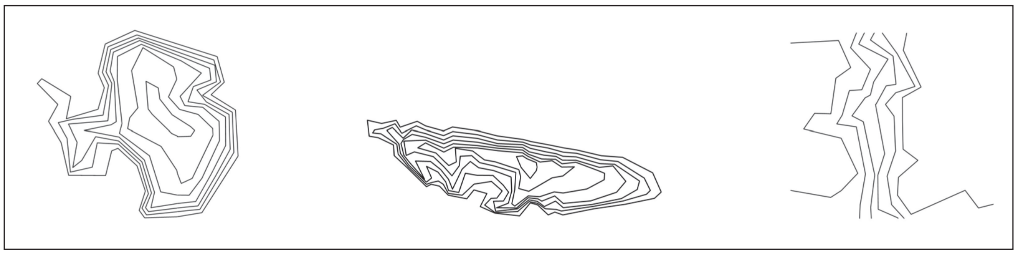


Figure 11. Harmonious simplifications of (left) AH1 with 150 vertices, (middle) AS2 with 250 vertices, (right) AS1 with 70 vertices

3.2. Quantitative Analysis

The aim of our algorithm is to maintain harmony as a visual cue that the isolines describe a single underlying object. In this section we check whether our algorithm is also measurably successful at maintaining harmony. We compare the results of our algorithm to a modified version in which each edge is treated “independently”. We perform the exact same collapsing procedure, but do so for only one edge at a time. We refer to this modified version as the *isolated algorithm*, contrasting our proposed *harmonious algorithm*.

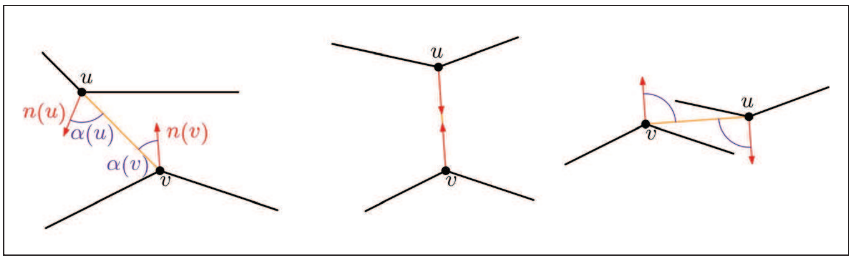


Figure 12. (left) The harmony measure for a matched pair (*u, v*) is the sum of the angles indicated in purple ((*u*) + (*v*)). Note that the direction of the normal is irrelevant for the score. (middle) A “perfect” pair, scoring zero. (right) A “worst” pair, scoring 180 degrees

Measuring Harmony. To support a quantitative comparison, we must quantify the harmony of a set of isolines. We use a simple measure that captures one of the facets of harmony. Particularly, we compute the GLCFM between each subsequent pair of isolines and then measure the harmony for each matched pair. Intuitively, a pair of vertices (*u, v*) is matched harmoniously if the segment *uv* is locally orthogonal to the isolines. Thus, we estimate a normal *n*(*u*) at vertex *u* and a normal *n*(*v*) at *v*. We then measure the minimal angle (*u*) (respectively (*v*)) between the line spanned by *uv* and *n*(*u*) (respectively *n*(*v*)) (see Fig. 12). The sum of (*u*) and (*v*) is the measured harmony for this pair. This score ranges from zero (“perfect” harmony) to 180 degrees (“worst” harmony).

Our straight harmony line enables good harmony across multiple isolines: these vertices are expected to be matched in the eventual GLCFM and a vertex’s normal can be close to the matching lines to both its neighbors only if these lines have similar orientation.

Measuring Similarity. Typical approaches to simplification quantify the quality of the resulting simplification by its (geometric) similarity to the input. We may hence generally use geometric similarity as a measure of quality; for our evaluation we use the *continuous* Fréchet distance between a computed simplified isoline and its original geometry for this purpose. We normalize this distance between instances using the diagonal of the bounding box of the input. We use the continuous version, as the strongly differing complexities between the two polygonal lines causes the discrete version to not measure shape similarity very well.

Setup. We track how harmony develops throughout both algorithms as complexity decreases. We first run our harmonious algorithm until it is stuck, saving intermediate results when the complexity first reaches or drops below one of a given sequence of desired complexities[2]. An example sequence obtained in this manner is shown in Figure 13. Since a collapse may reduce the total complexity by the complexity of the ladder, the resulting maps do not necessarily have the exact desired complexities. We then run the isolated algorithm, using the same procedure but as desired complexities, those actually achieved by the harmonious algorithm.

We measure the harmony and similarity as both algorithms progress, that is, in their sequence of maps. We plot the average and maximum harmony over all matched pairs and average and maximum similarity over all isolines as a function of complexity (Figure 14).

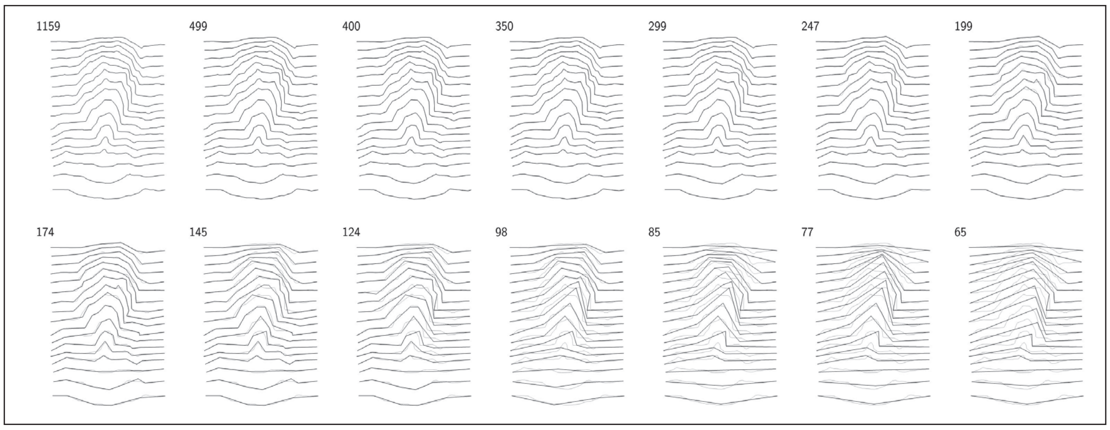
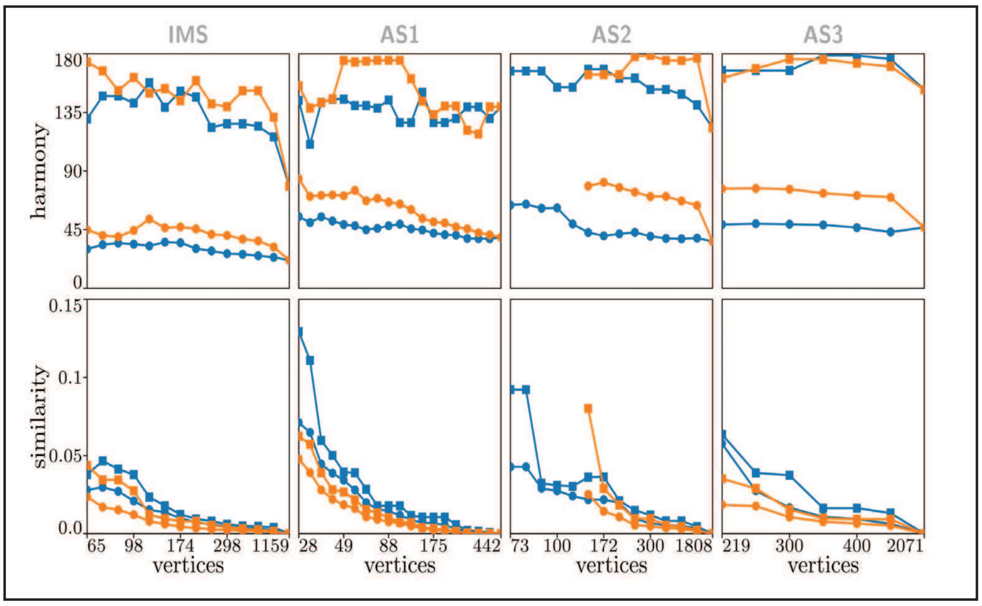


Figure 13. Results of IMS using our harmonious algorithm; numbers indicate complexity



[2]500, 400, 350, 300, 250, 200, 175, 150, 125, 100, 90, 80, 70, 60, 50, 45, 40, 35, 30, 25, 20

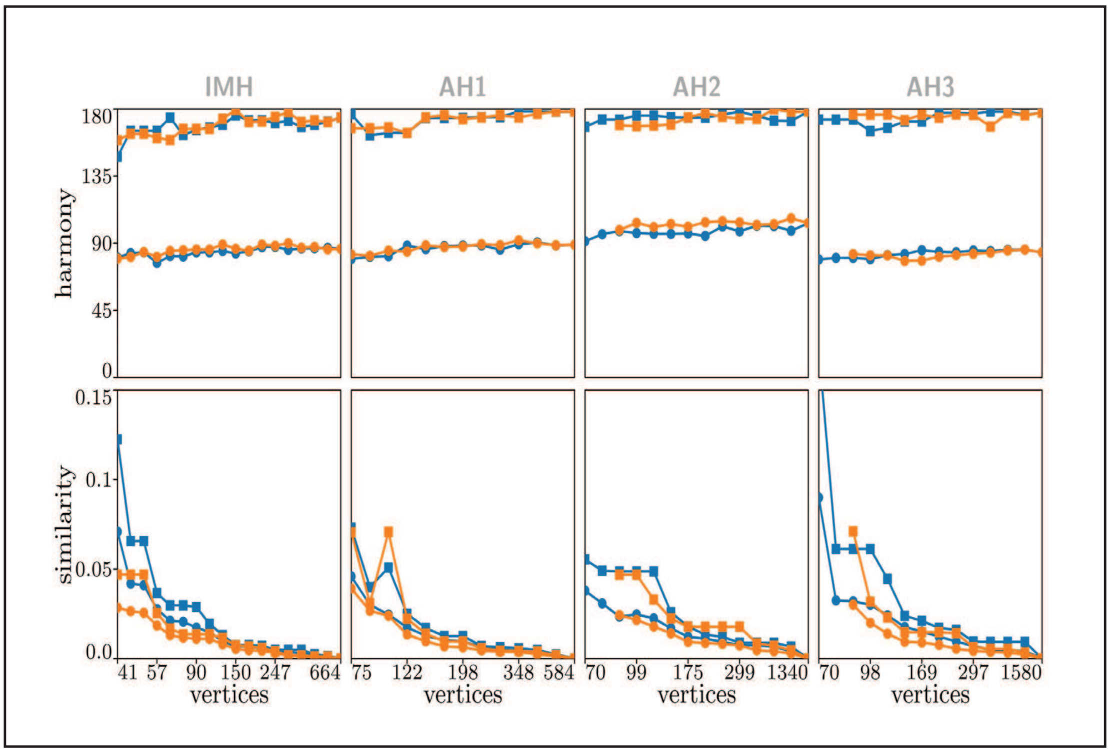


Figure 14. Average (circles) and maximum (squares) harmony and similarity as the algorithms simplify the input; lower values are better. Blue: harmonious algorithm; orange: isolated algorithm. Note that the horizontal axis is not linear. The maximum similarity (0.27) for the result of our harmonious algorithm on AH3 with 81 vertices has been cropped

Observations. First, we observe that the harmonious algorithm tends to perform better for harmony than the isolated method, but the similarity distance is typically slightly worse. This stands to reason as a collapse may sacrifice similarity to improve harmony. However, the effect on similarity is rather minor, until we reach the lower levels of complexity.

Second, the average and maximum Fréchet distance behave very similarly and tend to not differ much until the algorithm reaches the lowest complexities. This suggests that the geometric distortion our algorithm causes remains evenly spread between the isolines.

Third, we observe that for the isolated algorithm the average harmony tends to deteriorate. For the harmonious algorithm the harmony tends to stay roughly constant, though some of the slopes do show a bit of a deterioration as well (**AS1–AS2** mostly).

Fourth, we observe that the isolated algorithm often is stuck earlier, never reaching the number of vertices that the harmonious algorithm can reach; this is the case for **AS2–AS3**, **AH1–AS3**. This may seem somewhat counterintuitive: staying closer to the original geometry should generally help avoid intersections. There are two possible explanations here: (1) by collapsing entire slope ladders, we may still perform a collapse, even if collapsing any of the rungs in isolation would cause intersections; (2) by incorporating harmony, there is a better relation between the isolines and thus performing a collapse (even if in isolation) is less likely to cause intersections. We believe the first explanation is most likely the strongest effect here, but the second explanation may warrant further investigation in future work.

4. Conclusion and future work

We have presented an algorithm for simplifying isolines harmoniously, i.e., acknowledging that they describe a single object and that structures of this object should be reflected clearly in the result. Our evaluation shows the efficacy of our method, but also marks some current limitations to be overcome in future work. In addition to these, we observe that simplifying larger features (more than a single slope or hilltop) requires more investigation. For example, if a hill has two hilltops, we may need to match the isoline that encompasses both partially to one hilltop and partially to the other. In similar spirit, even between two isolines, large flat areas should possibly be omitted in the matching as maintaining harmony over a large distance is counter to the underlying principle; again, some form of partial matching may be useful here. As object identification in isoline systems is quite advanced and computationally efficient via contour trees and similar Reeb-graph-like structures, the starting and end-point problems for partial matchings within our method remain future work.

References

[1] Alt, Helmut., Behrends, Bernd., Blömer, Johannes. (1995). Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13 (3) 251–265.

[2] Alt, Helmut., Godau, Michael. (1995). Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5 (01n02), 75–91, 1995.

[3] Bellman, Richard., Kalaba, Robert. (1959). On adaptive control processes. *IRE Transactions on Automatic Control*, 4 (2) 1–9.

[4] Bose, Prosenjit., Cabello, Sergio., Cheong, Otfried., Gudmundsson, Joachim., Kreveld, Marc J. van., Speckmann, Bettina. (2006). Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, 4 (4) 554–566.

[5] Buchin, Kevin., Buchin, Maike., Meulemans, Wouter., Speckmann, Bettina. (2019). Locally correct Fréchet matchings. *Computational Geometry*, 76: 1–18.

[6] Buchin, Kevin., Meulemans, Wouter., Renssen, André Van., Speckmann, Bettina. (2016). Areapreserving simplification and schematization of polygonal subdivisions. *ACM Transactions on Spatial Algorithms and Systems*, 2 (1) Article No. 2, 1–36.

[7] Dijk, Thomas C. van., Goethem, Arthur van., Haunert, Jan-Henrik., Meulemans, Wouter., Speckmann, Bettin. (2014). A Map schematization with circular arcs. *In*: Proceedings International Conference on Geographic Information Science, pages 1–17.

[8] Regina Estkowski and Joseph S. B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. *In*: *Proceedings 17th Symposium on Computational Geometry*, pages 40–49.

[9] Gaffuri, Julien., Duchêne, Cécile., Ruas, Anne. (2008). Object-field relationships modelling in an agent-based generalisation model. *In*: Proceedigs 12th Workshop on Generalisation and Multiple Representation.

[10] Gao, Aji., Li, Jingzhong., Chen, Kai. (2020). A morphing approach for continuous generalization of linear map features. Plos one, 15 (12) e0243328.

[11] Goethem, Arthur van., Meulemans, Wouter., Reimer, Andreas., Speckmann, Bettina. (2020). Simplification with parallelism. In : Proceedings 23rd ICA Workshop on Generalisation and Multiple Representation.

[12] Goethem, Arthur van., Meulemans, Wouter., Speckmann, Bettina., Jo Wood. (2015). Exploring curved schematization of territorial outlines. *IEEE Transactions on Visualization and Computer Graphics*, 21 (8) 889–902.

[13] Goldman, Alan H. (1990). Aesthetic qualities and aesthetic value. *The Journal of Philosophy*, 87 (1) 23–37.

[14] Guibas, Leonidas J., Hershberger, John. (1989). Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39 (2) 126–152.

[15] Guilbert, Eric. (2016). Feature-driven generalization of isobaths on nautical charts: A multi-agent system approach. *Transactions in GIS*, 20 (1) 126–143.

[16] Guilbert, Eric., Gaffuri, Julien., Jenny, Bernhard. (2014). Terrain generalisation. In Abstracting geographic information in a data rich world, LNCG, pages 227–258. Springer.

[17] Howat, Ian M., Porter, Claire., Smith, Benjamin E., Noh, Myoung-Jong., Morin, Paul. (2019). The reference elevation model of Antarctica. *The Cryosphere*, 13 (2) 665–674.

[18] Imhof, Eduard. (1965). Kartographische Geländedarstellung. De Gruyter.

[19] Kronenfeld, Barry J., Stanislawski, Lawrence V., Buttenfield, Barbara P., Brockmeyer, Tyler. (2020). Simplification of polylines by segment collapse: Minimizing areal displacement while preserving area. *International Journal of Cartography*, 6 (1) 22–46.

[20] Li, Zhilin., Sui, Haigang. (2000). An integrated technique for automated generalization of contour maps. *The Cartographic Journal*, 37 (1) 29–37.

[21] Löffler, Maarten., Meulemans, Wouter. (2017). Discretized approaches to schematization. *In*: Proceedings 29*th* *Canadian Conference on Computational Geometry*, pages 220–225.

[22] Mendel, Thomas. (2018). Area-preserving subdivision simplification with topology constraints: Exactly and in practice. In Proceedinga 20*th* Workshop on Algorithm Engineering and Experiments, pages 117–128.

[23] Paulo Raposo. (2013). Scale-specific automated line simplification by vertex clustering on a hexagonal tessellation. *Cartography and Geographic Information Science*, 40 (5) 427–443.

[24] Andreas Reimer. (2015). Cartographic modelling for automated map generation. PhD thesis, Technische Universiteit Eindhoven.

[25] Samsonov, Timofey, E. (2020). Automated conflation of digital elevation model with reference hydrographic lines. *ISPRS International Journal of Geo-Information*, 9 (5).

[26] Samsonov, Timofey E., Koshel, Sergey., Walther, Dmitry., Jenny, Bernhard. (2019). Automated placement of supplementary contour lines. *International Journal of Geographical Information Science*, 33(10) 2072–2093.

[27] Schüle, Wilhelm., Zur Maßstabsfrage des neuen schweizerischen Kartenwerkes, mit einem Nachtrag und Anhang zur Kurvendarstellung auf topographischen Karten. In Jahresbericht der Geographischen Gesellschaft von Bern, Bd. XXVIII, pages 31–53, 1929.

[28] Skopeliti., Andriani., Tsoulos., Lysandros., Pe'eri, Shachak. (2021). Depth contours and coastline generalization for harbour and approach nautical charts. *ISPRS International Journal of Geo-Information*, 10 (4) 197.

[29] Tong, Xiaohua., Jin, Yanmin., Li, Lingyun., Ai, Tinghua. (2015). 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.

[30] Touya, Guillaume., Boulze, Hugo., Schleich, Anouk., Quinquenel, Hervé. (2019). Contour lines generation in karstic plateaus for topographic maps. *In*: Proceedings *International Cartographic Conference*, volume 2, pages 1–8.

[31] Tutic, Dra•en., and Lapaine, Miljenko. (2009). Area preserving cartographic line generalizaton. Kartografija i geoinformacije (Cartography and Geoinformation), 8 (11) 84–100.

[32] Tutic, Dra•en, Štanfel, Matja•., Jogun, Tomislav. (2017). Automation of cartographic generalisation of contour lines. *In*: Proceedings 10th ICA Mountain Cartography Workshop, pages 65–77.