Morphing Polylines Based on Least-Squares Adjustment

Dongliang Peng[1,3], Jan-Henrik Haunert[1], Alexander Wolff[1], Christophe Hurter[2]

[1]Institute of Computer Science, University of Würzburg, Germany
http://www1.informatik.uni-wuerzburg.de/en/staff

[2]ENAC/University of Toulouse, France
christophe.hurter@enac.fr

[3]Department of Geo-Informatics, Central South University, China

1. Introduction

Digital maps such as Google Maps or Open Street Map have become one of the most important sources of geographic information. When users interactively browse through such maps on computers or small displays, they often need to zoom in and out to get the information desired. Often, zooming is supported by a multiple representation database (MRDB). This stores a discrete set of levels of detail (LOD) from which a user can query the LOD for a particular scale (Hampe et al 2004). A small set of LODs leads, however, to large and sudden changes during zooming. Since this distracts users, hierarchical schemes have been proposed that implement the generalization process based on small incremental changes, for example, the BLG-tree (van Oosterom 2005) for line simplification or the GAP-tree (van Oosterom 1995) for area aggregation. The incremental generalization process is represented in a data structure that allows a user to retrieve a map of any desired scale. Still, the generalization process consists of discrete steps and includes abrupt changes. To achieve a continuous generalization, Sester and Brenner (2004) suggested to simplify building footprints based on small incremental steps and to animate each step smoothly. Also aiming at a continuous generalization, several authors have developed methods for morphing between two polylines (Cecconi 2003, Nöllenburg et al. 2008). Most of these methods consist of two steps (Cecconi 2003, Nöllenburg et al. 2008, Peng et al. 2012). The first step usually identifies the corresponding vertices of the two polylines. The second step defines a trajectory for each pair of corresponding vertices. Most often, straight-line trajectories are defined on which, when morphing, the vertices move at constant speed.

In this paper we address morphing, but we relax the requirement that the vertices of the polyline move on straight lines. Our concern with straight-line trajectories is that characteristic properties of the polylines change drastically during the morphing process. In particular, we suggest that the angles and edge lengths of the polyline should change linearly during the morphing process. As Figures 1(a) and 1(b) show, this is clearly not accomplished with straight-line trajectories. In contrast, the new method that we present in this paper yields a close-to-linear relationship, for example, between time and edge lengths; see Figures 1(c) and 1(d).

The paper is organized as follows. We review related work in Section 2. The details of our method are presented in Section 3, which include a list of soft and hard constraints, estimates for the unknown parameters, and the iterative process of our model. We present a case study in Section 4, which shows that our method generally performs well but also reveals new problems. We conclude the paper in Section 5.

2. Related Work

Different methods of morphing have been developed for maps. In map generalization, there are many constraints that need to be satisfied (Weibel and Dutton 1998, Harrie

1999). These constraints should also be satisfied by intermediate-scale features displayed when morphing. According to van Kreveld (2001), the amount of displacement between the corresponding vertices of two maps of different scales is quite small and, when using straight-line trajectories for morphing, hardly any features will be in conflict with the interpolated features. We, however, argue that even in simple situations such as the one in Figures 1(a) and 1(b) straight-line trajectories fail to generate satisfactory intermediate-scale features. In fact, there are methods that use curves rather than straight lines as vertex trajectories, for example, circular arcs or parabolas (Whited and Rossignac 2009). In contrast to these methods, our method does not require the trajectories to be of any particular curve type. Instead, we define the morphing process based on constraints that we impose on the features at intermediate scales.

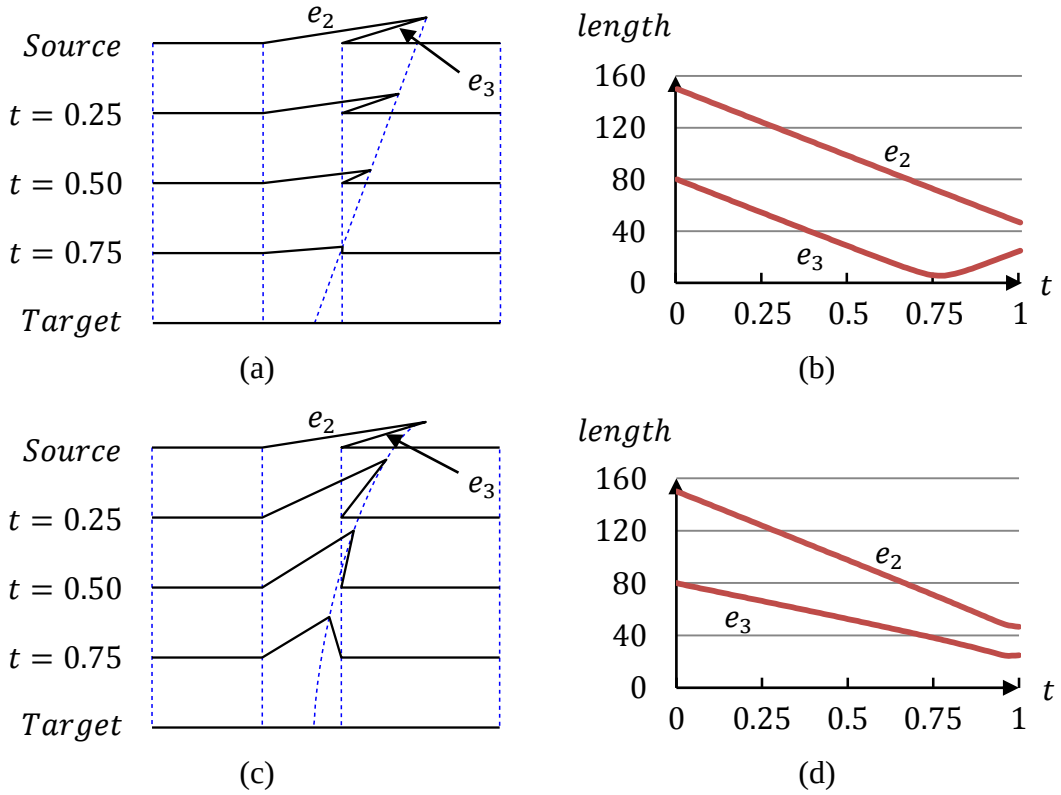


Figure 1. Morphing between a source and a target polyline. When morphing based on straight-line trajectories (a), edge e_3 receives almost zero length at time $t = 0.75$ and then grows again (b). With our method (c) edge lengths change almost linearly (d).

Intermediate features are expected to be similar to the source feature and the target feature. We consider the angles and edge lengths to be very important attributes of a feature, not least because similarity measures are often defined based on the angles and edge lengths (e.g. Arkin et al. 1991, Latecki and Lakämper 2000, Frank and Ester 2006). Sederberg et al. (1993) proposed to morph two polygons by changing the angles and edge lengths linearly over time. The authors also showed how to tweak the edge lengths and/or the angles to guarantee that at any time the intermediate polygon is closed. We choose an approach similar to Sederberg et al., that is, we also try to achieve that the angles and edge lengths change linearly. Unlike Sederberg et al., however, we simultaneously handle multiple constraints by defining (and solving) the model of a least-squares adjustment. A completely different approach was taken by Connelly et al. (2003). They proved that any polygonal line can be *straightened*, that is,

its vertices can be moved to a straight line such that edge lengths never change and edges never intersect. Streinu (2000) showed that a quadratic number of moves suffices and how to compute those. Note, however, that, in order to morph a polygonal line A into another polygonal line B (with the same edge lengths), these procedures would morph A into a straight line and then into B, thereby changing angles arbitrarily.

Least-squares adjustment has been shown to be effective in handling multiple constraints in map generalization methods (Sester 2000, Harrie and Sarjakoski 2002). Basically, it relies on a function $\phi: \mathrm{R}^u \rightarrow \mathrm{R}^n$ that defines the relationship between a vector $\hat{X}$ of u unknowns and a vector L of n observations. Given the function ϕ and the vector L, it is reasonable to ask for a vector $\hat{X}$ strictly satisfying $L = \phi(\hat{X})$. Such a vector, however, normally does not exist since n is usually larger than u. Therefore, the vector of corrections v is introduced. Then, it is aimed to find $\hat{X}$ and v such that

$$L + v = \phi(\hat{X}), \tag{1}$$

and $v^T P v$ is minimal, where P is a matrix that allows different weights to be set to different observations.

Least-squares adjustment is particularly easy to solve if a linear relationship between the unknowns and the observations exists, that is, if $\phi(\hat{X}) = A\hat{X} + d$, where A is a constant matrix and d is a constant vector. An optimum solution is given with

$$\hat{X} = X_0 + (A^T P A)^{-1} A^T P l, \tag{2}$$

where X_0 can be any vector of dimension u and $l = L - \phi(X_0)$.

If there is no linear relationship between the unknowns and the observations, X_0 has to be chosen close to the optimum solution and the matrix A is defined based on the partial derivatives of ϕ at point X_0. Usually, Equation 2 then yields an approximation of the optimum unknown vector that is better than X_0. A good estimate can be found by iteratively solving Equation 2. In each iteration (except the first), the vector X_0 is set to the vector $\hat{X}$ found in the previous iteration.

Since eighty percent of all objects (points, lines, and areas) in a typical medium-scale topographic map consist of lines (United Nations 1989, cited in Muller 1991), we focus on this object type and propose a morphing method for polylines.

3. Morphing Based on Least-Squares Adjustment

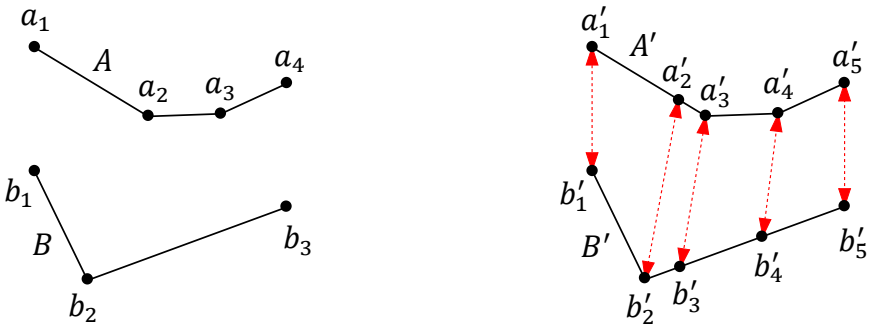
In this section, we present our new morphing method for polylines based on least-squares adjustment. We introduce some definitions in Section 3.1. Then, multiple requirements are modeled as constraints. The soft constraints are presented in Section 3.2. We set constant values to the coordinates of some vertices to implement hard constraints in Section 3.3. The estimates for the unknowns are given in Section 3.4. Finally, we sketch the stop condition of the model in Section 3.5.

3.1 Preliminaries

We assume that we are given a polyline A with vertices $a_1, \ldots, a_M$ and a polyline B with vertices $b_1, \ldots, b_N$, where A and B represent the same geographic feature. Vertices a_1 and b_1 as well as a_M and b_N correspond to each other; see Figure 2(a). For every vertex of A we are given a corresponding point (not necessarily a vertex) on B and for every vertex of B we are given a corresponding point (not necessarily a vertex) on A. The points corresponding to $a_1, \ldots, a_M$ are ordered along B and the points

corresponding to $b_1, \dots, b_N$ are ordered along A. How to find the corresponding pairs of vertices is not discussed in this paper. We apply an algorithm similar to that of Nöllenburg et al. (2008) to solve this task, but any other method could be used as well.

In order to ensure that both polylines have the same number n of vertices, our method first injects the points corresponding to $a_1, \dots, a_M$ as new vertices into B (yielding polyline B') and the points corresponding to $b_1, \dots, b_N$ as new vertices into A (yielding polyline A'), as shown in Figure 2(b). The morphing process starts at time 0 with polyline A' and ends at time 1 with polyline B'. Generally, we denote the polyline that is displayed at time t by $P(t)$, thus $P(0) = A'$ and $P(1) = B'$, and require that $P(t)$ has n vertices. We denote the i-th vertex of $P(t)$ by $p_i(t)$.



(a) original polylines (b) polylines after injection of vertices

Figure 2. Illustration of corresponding vertices.

3.2 Soft constraints

The polyline $P(t)$ at time t is computed by constraining its angles and the lengths of its edges. That is, for each angle and edge length, we define an observation, which can be interpreted as an ideal value that we would like to achieve. In most cases, these ideal values can't be achieved at the same time because of some necessary hard constraints. Therefore, we apply least-squares adjustment, which allows a polyline to be computed such that its actual angles and edge lengths are as close as possible to the observations. More precisely, the square sum of the differences between the observations and the actual parameters of the polyline are minimized. These angles and edge lengths constitute soft constraints in our method.

In order to achieve that the polyline behaves as in our motivating example (recall Figures 1(c) and 1(d)), we define each observation based on a linear interpolation between the initial and final value of the polyline parameter corresponding to that observation.

For the angles, this means that

$$\beta_i(t) = (1 - t) \cdot \beta_i(0) + t \cdot \beta_i(1), \tag{3}$$

where $i = 2, \dots, n - 1$, $\beta_i(0)$ and $\beta_i(1)$ are the initial angle and the final angle in the i-th vertex of A' and B', respectively, and $\beta_i(t)$ is the observation of the angle in the i-th vertex at time t; see Figure 3.

Similarly, for the edge lengths, we define

$$l_i(t) = (1 - t) \cdot l_i(0) + t \cdot l_i(1), \tag{4}$$

where $i = 1, \dots, n-1$, $l_i(0)$ and $l_i(1)$ are the initial length and the final length of the i-th edge of A' and B', respectively, and $l_i(t)$ is the observation of the length of the i-th edge at time t; see Figure 3.

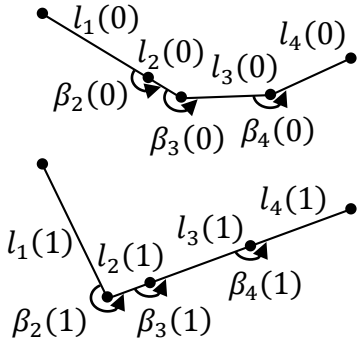


Figure 3. Illustration of initials and finals.

In order to apply least-squares adjustment, we have to express the relationship between the adjusted coordinates $\hat{x}_1(t), \dots, \hat{x}_n(t)$ and $\hat{y}_1(t), \dots, \hat{y}_n(t)$ of the vertices of $P(t)$ and the observations. The adjusted coordinates $\hat{x}_1(t), \dots, \hat{x}_n(t)$ and $\hat{y}_1(t), \dots, \hat{y}_n(t)$ are the unknowns for the least-squares adjustment at time t. We first consider angles, then edge lengths.

Angles: An angle $\beta_i(t)$ can be computed by considering the difference of the two angles that the edges e_{i-1} and e_i form with the x-axis. Depending on the quadrants in which e_{i-1} and e_i lie relative to the vertex $p_i(t)$, a multiple of π has to be added. This means that, for the adjusted value $\hat{\beta}_i(t)$ of $\beta_i(t)$, we require that

$$\hat{\beta}_i(t) = \arctan\frac{\hat{y}_{i+1}(t)-\hat{y}_i(t)}{\hat{x}_{i+1}(t)-\hat{x}_i(t)} - \arctan\frac{\hat{y}_i(t)-\hat{y}_{i-1}(t)}{\hat{x}_i(t)-\hat{x}_{i-1}(t)} + K_i \cdot \pi, \tag{5}$$

where $K_i \in \mathbb{Z}$ is a constant that only depends on i. For the least-squares adjustment, we have to compute the partial derivatives of $\hat{\beta}_i(t)$ with respect to the unknowns. Note that these do not depend on the constant term $K_i \cdot \pi$. Therefore, we can neglect it.

Edge lengths: Each edge length is a Euclidean distance. Hence, for the adjusted edge length $\hat{l}_i(t)$, we require

$$\hat{l}_i(t) = \sqrt{\left(\hat{x}_{i+1}(t) - \hat{x}_i(t)\right)^2 + \left(\hat{y}_{i+1}(t) - \hat{y}_i(t)\right)^2}. \tag{6}$$

Here, $\hat{\beta}_i$ and $\hat{l}_i$ constitute the function $\phi(\hat{x})$ which we introduced in Equation 1. Since $\hat{\beta}_i$ and $\hat{l}_i$ are not linear, we linearize them with respect to their partial derivatives (Sester 2000, Harrie and Sarjakoski 2002).

Note that without adding hard constraints to our model, there is no need for an adjustment, because we could perfectly satisfy every soft constraint, simply by creating a polyline with the desired angles and edge lengths. This will change, however, if we add hard constraints, for example, if we prescribe the end vertices of the polyline.

3.3 Hard constraints

There may be some common characteristic vertices in the polylines A' and B' which represent the same feature at different scales. These common characteristic vertices should keep their positions when morphing. That is, if for some $i \in \{1, \dots, n\}$ it holds

that $p_i(0) = p_i(1)$, we require as a hard constraint that for any $0 < t < 1$ it also holds $p_i(0) = p_i(t) = p_i(1)$. That is to say, for the coordinates of $p_i(t)$, we do not introduce unknown parameters in the least-squares adjustment. Note that our method does not require the existence of such common characteristic vertices, but it can handle them.

Even if a vertex $p_i(0)$ in A' does not have the exact same position as its corresponding vertex $p_i(1)$ in B', we may want to constrain $p_i(t)$ to lie at a prescribed position. In particular, by prescribing the positions of the end vertices $p_1(t)$ and $p_n(t)$ of the polyline $P(t)$, we can compute $P(t)$ independently of other polylines that need to meet $P(t)$ in one of its end vertices. This is useful if we need to deal with a geometric graph that, for example, represents a road network. We suggest prescribing the positions of vertices of degree higher than two (that is, road junctions), which allows us to treat each path between two such vertices as an independent problem instance. When prescribing the position of a vertex $p_i(t)$, we apply a simple linear interpolation between $p_i(0)$ and $p_i(1)$, that is, we set

$$\begin{pmatrix} x_i(t) \\ y_i(t) \end{pmatrix} = (1-t) \cdot \begin{pmatrix} x_i(0) \\ y_i(0) \end{pmatrix} + t \cdot \begin{pmatrix} x_i(1) \\ y_i(1) \end{pmatrix}, \tag{7}$$

where $x_i(t)$ and $y_i(t)$ are the x- and y-coordinates of $p_i(t)$. Obviously, we should not constrain too many vertices this way, since otherwise no improvement compared to the existing method based on straight-line trajectories would be achieved.

We note that additional hard constraints are needed, for example, to ensure that some right angles remain during morphing. This constraint, however, is currently not considered in our method.

3.4 Estimates

To define the morphing process, we compute intermediate polylines for k values of the time parameter t. Choosing k large enough gives a smooth animation. We define each step to take the same amount of time, thus in the i-th step the time parameter t is $i/(k+1)$. We compute the polylines $P(1/(k+1)), P(2/(k+1)), \ldots, P(k/(k+1))$ in succession. Since the polyline at time $i/(k+1)$ will be similar to the polyline at time $(i-1)/(k+1)$, we use the vertex coordinates of the previously computed polyline as estimates for the unknowns in the least-squares adjustment.

3.5 Iterative process

Since our model contains non-linear constraints, we need to solve it iteratively. We stop the iteration process as soon as the norm of vector $\hat{x}(t)$ is smaller than a user-set threshold, where $\hat{x}(t) = (A^T P A)^{-1} A^T P l$.

4. Case Study

To get reasonable correspondence relationships for two polylines that will be morphed, we used a dynamic-programming algorithm similar to that of Nöllenburg et al. (2008) to determine the pairs of corresponding vertices. The algorithm of Nöllenburg et al. uses characteristic vertices (in our experiments, all the vertices are regarded as characteristic vertices) and segments between consecutive characteristic vertices as elements to match to minimize a defined cost function. To ensure the soft constraints of angles, we always prescribe the first two vertices and the last two vertices of the polylines in the least-squares adjustment.

4.1 Case study on artificial data

We tested our method on an instance by Bereg (2005); see Figure 4(a). The corresponding vertices determined by the dynamic-programming algorithm are shown in Figure 4(b).

(a) two polylines (b) corresponding vertices

Figure 4. A data set used in our experiments.

Figure 5 shows the results of morphing A to B based on straight-line trajectories and least-squares adjustment. Based on straight-line trajectories, the left part of the "bend" shrinks step by step and a self-intersection occurs at time $t = 0.75$. Based on least-squares adjustment, the same part of the "bend" firstly translates to the right side and then morphs to the target edges. Definitely, the latter results are more reasonable.

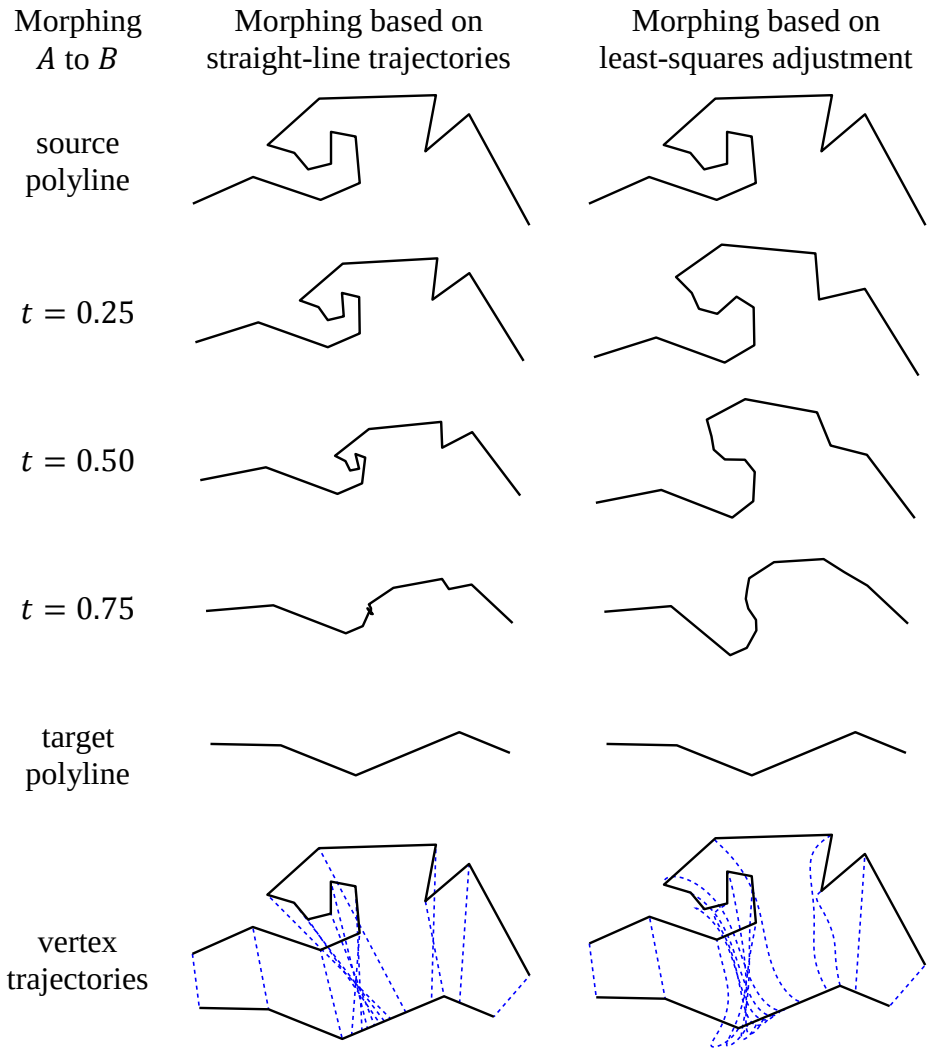
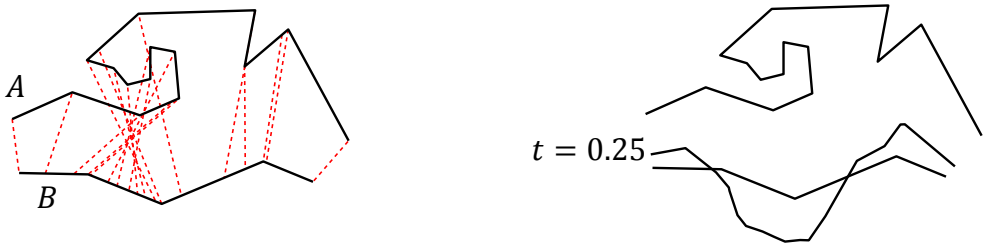


Figure 5. Morphing with straight-line trajectories (left) and with our method (right).

Unfortunately, there are still problems with our method. First, we encountered a problem when we defined the corresponding vertices of the polylines in Figure 4(a) with a simple linear interpolation algorithm (as shown in Figure 6(a)) rather than with the dynamic-programming algorithm. Morphing B to A by the least-squares adjustment method we obtained undesirable results (for example, at time $t = 0.25$; see Figure 6(b)). The polyline B moved down rather than up.

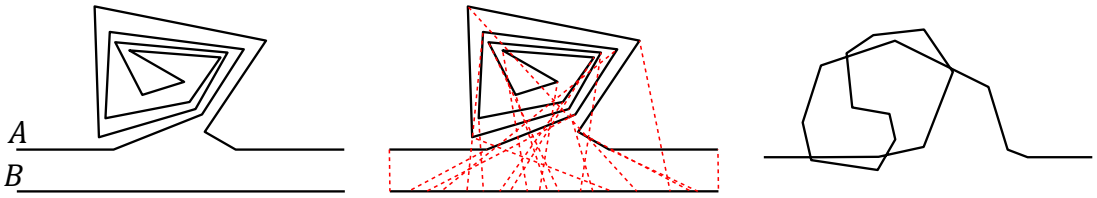
Second, because there is no sophisticated technique to avoid self-intersections in our method, self-intersections may be introduced as shown in Figure 7.

Third, sometimes the iterative computation doesn't converge if the polyline contains very short line segments. Figure 8 shows such a strange result. For the corresponding vertices shown in Figure 7(b), we add an extra pair of corresponding vertices (linked by a dashed green thicker curve; see Figure 8(a)) which is very close to one of the pairs of corresponding vertices, so that a pair of very short corresponding segments is introduced. The strange polyline generated with the extra vertices at time $t = 0.83$ is shown in Figure 8(b), where the green circle represents the extra vertex. By comparison, the polyline generated without the extra vertices is shown in Figure 8(c), where the green triangle represents the vertex near the extra vertex. Moreover, the algorithm doesn't converge at time $t = 0.90$ with the extra vertices.



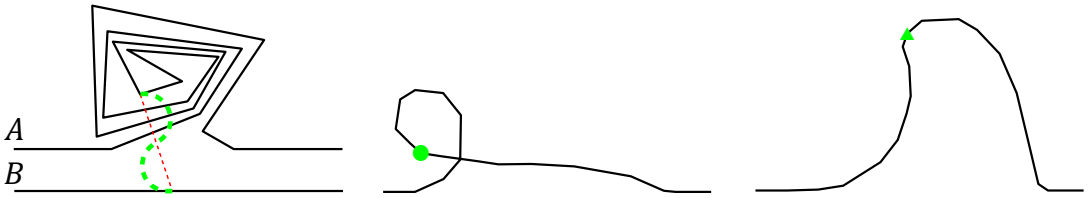
(a) corresponding vertices (b) morphing B to A at time t = 0.25

Figure 6. An example of an undesirable result based on our least-squares adjustment.



(a) input polylines (b) corresponding vertices (c) output for $t = 0.5$

Figure 7. A self-intersection generated by our least-squares adjustment.

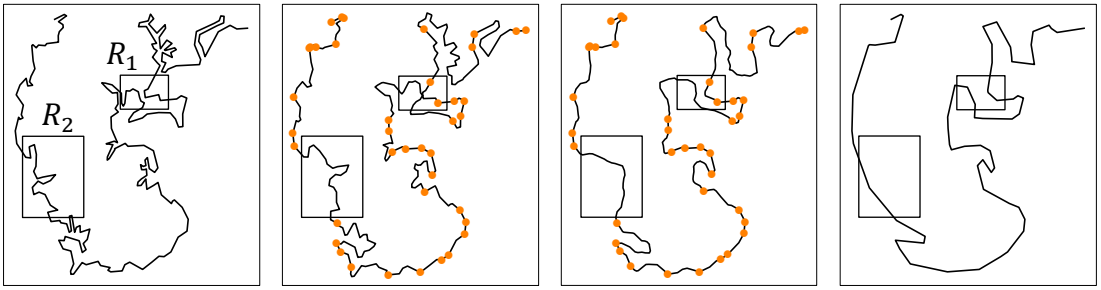


(a) the extra pair of corresponding vertices (b) $t = 0.83$, with the extra corresponding vertices (c) $t = 0.83$, without the extra corresponding vertices

Figure 8. An example of a strange result based on our least-squares adjustment.

4.2 Case study on real data

We tested our method on a part of the coastline of China; see Figure 9(a) and 9(d). The scale of the source polyline is $1:5{,}000{,}000$, the length is $1{,}002\ km$, and the number of vertices is 233; the scale of the target polyline is $1:30{,}000{,}000$, the length is $605\ km$, and the number of vertices is 66. The morphing results at times $t = 0.25$ and $t = 0.75$ are shown in Figure 9(b) and 9(c), respectively, where the prescribed vertices are marked by orange circles. Overall, we got nice results, but there are still some problems. In region R_1, the two segments almost intersect at time $t = 0.25$; in region R_2, the "bend" first expands and then shrinks, which is not reasonable. There are two reasons for both problems. First, the change (decrease or increase) of the angles is faster than needed. Second, the decrease of the lengths is slower than needed. Both reasons tend to make a bend expand and then shrink. As a result, a better fitting model is needed to simulate the change of angles and edge lengths.



(a) source polyline (b) $t = 0.25$ (c) $t = 0.75$ (d) target polyline

Figure 9. Case study on real data.

5. Concluding Remarks

We have introduced a morphing method for polylines that tries to achieve that angles and edge lengths change linearly over time. Our approach is based on least-squares adjustment and allows soft and hard constraints to be handled. Our first results are promising. Still, there are open problems. In particular, we have to ensure that our method always converges to a good solution. We also aim to model more constraints, for example, to avoid self-intersections. Besides, a further topic is to combine morphing and simplification.

Acknowledgements

This research was partly supported by the China Scholarship Council (CSC) and Hunan Provincial Innovation Foundation for Postgraduates.

References

Arkin E M, Chew L P, Huttenlocher D P, Kedem K and Mitchell J S B, 1991, An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216.

Bereg S, 2005, An approximate morphing between polylines. *International Journal of Computational Geometry*, 15(2):193–208.

Cecconi A, 2003, Integration of cartographic generalization and multi-scale databases for enhanced web mapping. Dissertation, University of Zurich.

Connelly R, Demaine E D and Rote G, 2003, Straightening polygonal arcs and convexifying polygonal cycles. *Discrete and Computational Geometry*, 30(2):205–239.

Frank R and Ester M, 2006, A quantitative similarity measure for maps. In: Riedl A, Kainz W and Elmes G A (eds), *Progress in Spatial Data Handling – Proc. 12th International Symposium on Spatial Data Handling*, Springer-Verlag, Berlin, Germany, pages 435–450.

Hampe M, Sester M and Harrie L, 2004, Multiple representation databases to support visualisation on mobile devices. In: Altan O (eds), *Proc. 20th ISPRS Congress*, volume XXXV (Part B4), series *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Istanbul, Turkey, pages 135–140.

Harrie L, 1999, The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and Geographic Information Science*, 26(1):55–69.

Harrie L and Sarjakoski T, 2002, Simultaneous graphic generalization of vector data sets. *Geoinformatica*, 6(3):233–261.

Latecki L J and Lakämper R, 2000, Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190.

Muller J-C, 1991, Generalization of spatial databases. In: Maguire D J, Goodchild M F and Rhind D W (eds), *Geographical Information Systems: Principles and Applications*, Longman Scientific & Technical, Harlow, UK, pages 457–475.

Nöllenburg M, Merrick D, Wolff A and Benkert M, 2008, Morphing polylines: A step towards continuous generalization. *Computers, Environment and Urban Systems*, 32(4):248–260.

Peng D L, Deng M and Zhao B B, 2012, Multi-scale transformation of river networks based on morphing technology. *Journal of Remote Sensing*, 16(5):953–968.

Sederberg T W, Gao P, Wang G and Mu H, 1993, 2-D shape blending: an intrinsic solution to the vertex path problem. In: *Proc. 20th Annual Conference on Computer Graphics and Interactive Techniques*, New York, USA, pages 15–18.

Sester M, 2000, Generalization based on least squares adjustment. In: Fritsch D and Molenaar M (eds), *Proc. 19th ISPRS Congress*, volume XXXIII (Part B4), series *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Amsterdam, The Netherlands, pages 931–938.

Sester M and Brenner C, 2004, Continuous generalization for fast and smooth visualization on small displays. In: Altan O (eds), *Proc. 20th ISPRS Congress*, volume XXXV (Part B4), series *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Istanbul, Turkey, pages 1293–1298.

Streinu I, 2000, A combinatorial approach to planar non-colliding robot arm motion planning. In: *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS'00)*, Redondo Beach, USA, pages 443–453.

United Nations, 1989, Modern mapping techniques. *United Nations Inter-Regional Seminar, Honefoss, Norway*.

van Kreveld M, 2001, Smooth generalization for continuous zooming. In: *Proc. 20th International Cartographic Conference*, Beijing, China, pages 2180–2185.

van Oosterom P, 1995, The GAP-tree, an approach to on-the-fly map generalization of an area partitioning. In: Muller J C, Lagrange J P and Weibel R (eds), *GIS and Generalization: Methodology and Practice*, Taylor & Francis, London, UK, pages 120–132.

van Oosterom P, 2005, Variable-scale topological data structures suitable for progressive data transfer: the GAP-face tree and GAP-edge forest. *Cartography and Geographic Information Science*, 32(4):331–346.

Weibel R and Dutton G, 1998, Constraint-based automated map generalization. In: *Proc. 8th International Symposium on Spatial Data Handling*, Vancouver, Canada, pages 214–224.

Whited B and Rossignac J, 2009, B-morphs between b-compatible curves in the plane. In: *Proc. 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, New York, America, pages 187–198.