Morphing Polygonal Lines: A Step Towards Continuous Generalization

D. Merrick[1], M. Nöllenburg[2], A. Wolff[3], M. Benkert[2]

[1] School of Information Technologies, University of Sydney, NSW 2006, Australia
and National ICT Australia, Alexandria, NSW 1435, Australia
dmerrick@it.usyd.edu.au, www.it.usyd.edu.au/∼dmerrick

[2] Faculty of Informatics, Karlsruhe University, P.O. Box 6980,76128 Karlsruhe, Germany
{noelle, mbenkert}@iti.uka.de, i11www.iti.uni-karlsruhe.de/group

[3] Department of Mathematics and Computing Science, Technische Universiteit
Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
www.win.tue.nl/∼awolff

1 Introduction

Visualization of geographic information in the form of maps has been established for centuries. Depending on the scale of the map the level of detail of displayed objects must be adapted in a *generalization* process. Be it done manually or (semi-)automatically, generalization methods usually produce a map at a single target scale. This is a well-studied field, surveyed, for example, by Weibel and Dutton (1999).

In current, often web-based (Jones and Ware, 2005), geographic information systems users can interactively zoom in and out of the map, ideally at arbitrary scales and with smooth, continuous changes. However, current approaches are often characterized by a fixed set of scales or by simply zooming graphically without modifying map objects. To overcome these deficiencies *continuous* generalization methods are needed.

This paper studies an algorithm for continuously generalizing linear features like rivers or roads between their representations at two scales. Instead of line-simplification methods with a single target scale, we look into interpolating between a source and a target scale in a way that keeps the maps at intermediate scales meaningful. In computer graphics and computational geometry this interpolation process is known as *morphing* (Gomes *et al.*, 1999). Of specific interest are morphings that can deal with a certain amount of exaggeration and schematization such as reducing the number but increasing the size of road serpentines at the smaller scale. Our method first partitions the input polygonal line (polyline, for short) into characteristic segments and then defines distances between these segments. Based on those distances we compute an optimum morphing of the polyline segments at the two input scales using dynamic programming. We have implemented a prototype of the algorithm and compare its output with that of a simple linear morph.

2 Related work

Cecconi and Galanda (2002) study adaptive zooming for web applications with a focus on the technical implementation. They use the standard Douglas-Peucker line-simplification method to generalize linear features. While maps can be produced at arbitrary scales there is no smooth animation of the zooming. A set of continuous generalization operators is presented by van Kreveld (2001), including two simple algorithms for morphing a polyline to a straight line segment. Continuous generalization for building ground plans and typification of buildings is described by Sester and Brenner (2004).

Existing algorithms for the geometric problem of finding an optimal intersection-free geodesic morphing between two simple, non-intersecting polylines (Efrat *et al.*, 2001; Bespamyatnikh, 2002) cannot be applied here because the two input polylines intersect in general. Surazhsky and Gotsman (2001a,b) compute trajectories for intersection-free morphings of plane polygonal networks using compatible triangulations. Similarly, Erten *et al.* (2004) give an algorithm for intersection-free morphing of plane networks using a combination of rigid motion and compatible triangulations. However, those approaches require a given correspondence between network nodes. In the field of computer graphics, Cohen *et al.* (1997) match point pairs sampled uniformly along two (or more) parametric freeform curves. They compute an optimal correspondence of the points w.r.t. a similarity measure based on the tangents of the curves. The algorithm is similar to ours in that it also uses dynamic programming to optimize the matching, but it does not take into account the characteristic points of geographic polylines. Samoilov and Elber (1998) extend the method of Cohen *et al.* (1997) by eliminating possible self-intersections during the morphing.

3 Model and algorithm

In this paper, we consider the problem of morphing between two given polylines, each generalized at a different scale. Our algorithms to solve the problem can be extended in a straightforward manner to finding a series of morphs across many scales, by solving each pair of polylines in the problem independently. The same approach can be applied to two networks with identical topology.

The problem of morphing between two polylines is two-fold. Firstly, a correspondence must be found between points on the two lines. Secondly, trajectories that connect pairs of corresponding points must be specified. Here we focus on the correspondence problem and assume straight-line trajectories.

In addressing the correspondence problem, our goal is to match parts of each polyline that have the same semantics, e.g. represent the same series of hairpin bends in a road at two levels of detail. We wish to do this in a way that allows the *mental map* to be retained as much as possible. The mental map is the mental image a person builds of a diagram. Retention of the mental map is believed to be important in continuous understanding of

animated diagrams; see for example Misue *et al.* (1995). One important aspect of retaining the mental map is to ensure that points that are initially close together do not move too far apart. We therefore wish to minimize the movement of points from one polyline to another. To create a morph with these desired properties, we first detect characteristic points of a polyline (Section 3.1) and use these to find an optimum correspondence (Section 3.2).

Formally, we are given two polylines f and g in the plane $\mathbb{R}^2$. In the correspondence problem we need to find two continuous, monotone parameterizations $\alpha : [0,1] \rightarrow f$ and $\beta : [0,1] \rightarrow g$, such that $\alpha(0)$ and $\beta(0)$ map to the first points of f and g and $\alpha(1)$ and $\beta(1)$ map to the last points, respectively. These two parameterizations induce the correspondence between f and g: for each $u \in [0,1]$ the point $\alpha(u)$ is matched with $\beta(u)$.

3.1 Detection of characteristic points

In order to solve the correspondence problem, we first need to divide each polyline into subpolylines to be matched up. We do this by locating points on each line that are considered to be characteristic of the line; each of these characteristic points then defines the end of one subpolyline and the start of another.

Previous work on generalization notes the importance of inflection points, bend points, and start and end points in defining the character of a line (Plazanet *et al.*, 1995). To find such points, we process each of the vertices in a polyline in order, checking at each if the sign of curvature has changed (an inflection point) or if the vertex is a point of locally maximal curvature (a bend point). We also apply thresholding and Gaussian filtering techniques to minimize error on noisy or poorly sampled polylines, as detailed in Algorithm 1. Gaussian filtering is a method of smoothing curves often used to assist in analyzing noisy curves; Lowe (1989) gives further details and an efficient algorithm.

Algorithm 1: Characteristic point detection

Input: Polyline f, number of sample points n', Gaussian smoothing factor σ, threshold angles θ_i, θ_b and θ_c

Output: Set of characteristic points C

1. Resample f using n' equally-spaced points to create a new polyline f'
2. Apply an in-place Gaussian filter with factor σ to smooth f'
3. Mark inflection vertices with inflection angle $\geq \theta_i$
4. Mark bend vertices with bend angle between adjacent edges $\geq \theta_b$ and change in curvature $\geq \theta_c$ from last point of locally minimal curvature
5. Mark first and last vertices
6. Proceed through the smoothed polyline f' and store the distance of each marked vertex from the start of f' as a percentage of the length of f'
7. Let C be the set of points at the stored percentage distances along the original polyline f

Algorithm 1 requires $O(|f| + n')$ time and space, where $|f|$ is the number of vertices on the polyline f. All input parameters are user-defined. Their values influence the number of characteristic points that will be detected. The only parameter that affects the running time is n', the number of sample points. See the results in Section 4 for sample values.

3.2 Finding an optimum correspondence

We detect the characteristic points of f and g independently of each other. Assume that there are $n + 1$ such points on f and $m + 1$ points on g, which divide the polylines into two sequences of subpolylines $(f_1, \ldots, f_n)$ and $(g_1, \ldots, g_m)$. Next, we approach the correspondence problem. Basically, there are five possibilities to match a subpolyline f_i:

(a) f_i is mapped to the last characteristic point g_j^{last} of a subpolyline g_j (i.e., f_i disappears),

(b) a subpolyline g_j is mapped to the last point f_i^{last} of f_i (i.e., g_j disappears),

(c) f_i is mapped to a subpolyline g_j,

(d) f_i is mapped to a merged polyline $g_{j\ldots(j+k)}$, and

(e) f_i is part of a merged polyline $f_{\ell\ldots i\ldots(\ell+k)}$ that is mapped to a subpolyline g_j.

Clearly, the linear order of the subpolylines along f and g has to be respected by the assignment.

Now assume that there is a *morphing cost* δ associated with the morph between two polylines. We suggest a morphing distance in the next section, but Algorithm 2 is independent of the concrete distance. It is based on dynamic programming and computes a minimum-cost correspondence. Algorithm 2 recursively fills an $n \times m$ table T, where the entry $T[i, j]$ stores the minimal cost of morphing $f_{1\ldots i}$ to $g_{1\ldots j}$. Consequently, we can obtain the optimum correspondence from $T[n, m]$.

The required storage space and running time of Algorithm 2 is $O(nm)$ provided that the *look-back parameter* K is constant. This parameter determines the maximum number of subpolyline segments that can be merged in order to match them with another segment in cases (d) and (e). Otherwise the running time increases to $O(nm(n + m))$.

Distance measure. Algorithm 2 relies on a distance function δ that represents the morphing cost of a pair of polylines. Distance functions for polylines can be defined in many ways, e.g. *morphing width* (Efrat *et al.*, 2001) and *Fréchet distance* (Alt and Godau, 1995).

We define a new distance measure that takes into account how far *all* points move during the morphing by integrating over the trajectory lengths. Assume that two subpolylines f_i and g_j with uniform parameterizations α and β are given. Each point $\alpha(u)$ on p will move

Algorithm 2: Optimum correspondence

Input: polylines $f = (f_1, \ldots, f_n)$ and $g = (g_1, \ldots, g_m)$, distance matrix δ
Output: optimum correspondence for f and g

initialize $T[0, \cdot]$ and $T[\cdot, 0]$
for $i = 1$ **to** n **do**
  for $j = 1$ **to** m **do**

$$T[i,j] = \min \begin{cases} T[i-1,j] + \delta(f_i, g_j^{\text{last}}) & \text{case (a)} \\ T[i,j-1] + \delta(f_i^{\text{last}}, g_j) & \text{case (b)} \\ T[i-1,j-1] + \delta(f_i, g_j) & \text{case (c)} \\ T[i-1,j-k] + \delta(f_i, g_{(j-k+1)\ldots j}),\ k = 2, \ldots, K & \text{case (d)} \\ T[i-k,j-1] + \delta(f_{(i-k+1)\ldots i}, g_j),\ k = 2, \ldots, K & \text{case (e)} \end{cases}$$

    store pointer to predecessor, i.e., to the table entry that yielded the minimum
  end
end
generate optimum correspondence from $T[n, m]$ using backtracking along pointers

to $\beta(u)$ on q along the connecting segment of length $||\alpha(u) - \beta(u)||$. Then the morphing distance is defined as

$$\delta(f_i, g_j) = \int_0^1 ||\alpha(u) - \beta(u)|| du \tag{1}$$

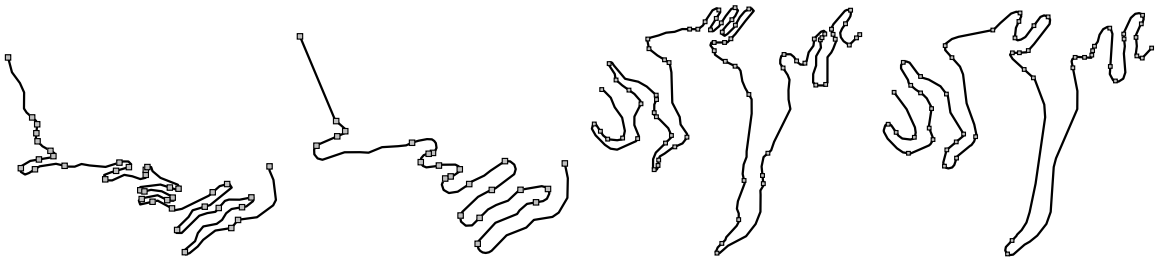
and can be computed in time linear in the complexity of f_i and g_j.

Optionally, we can add further terms to the base distance δ. Adding the length difference of f_i and g_j, or alternatively the length of the polyline $\gamma(u) := \alpha(u) - \beta(u)$ favors pairs of polylines that are roughly the same length or orientation. We can also multiply δ by the ratio of the subpolylines' length with the total length of the containing polylines f and g, to account for their relative visual importance.

Finally, we wish to avoid self-intersections in the morph. We do this locally by setting the effective morphing distance to ∞ if matching two subpolylines causes a self-intersection in the morph between them. However, in rare cases intersections between two non-corresponding subpolylines may still occur.

4 Results

We ran our implementations on a small set of French roads from the BD Carto® and the TOP100 series maps produced by the IGN Carto2001 project (Lecordix *et al.*, 2005). For each road, we used a polyline from BD Carto® at scale 1:50,000, and a generalized version at scale 1:100,000 from the Carto2001 TOP100 maps. Figures 1(a) and 1(b) show two examples of the roads in the dataset, at the two respective scales. The characteristic



(a) Road 1 (1:50,000) (b) Road 1 (1:100,000) (c) Road 2 (1:50,000) (d) Road 2 (1:100,000)

Figure 1: Example roads at two scales with detected characteristic points marked.

Polyline	$\mathbf{n}$	$\mathbf{n'}$	σ	$\theta_\mathbf{i}$	$\theta_\mathbf{b}$	$\theta_\mathbf{c}$
Road 1 (1:50,000)	135	600	line length / 400	$45°$	$8°$	$10°$
Road 1 (1:100,000)	118	600	line length / 400	$45°$	$5°$	$6°$
Road 2 (1:50,000)	255	600	line length / 800	$55°$	$10°$	$15°$
Road 2 (1:100,000)	202	600	line length / 800	$55°$	$6°$	$11°$

Table 1: Number of vertices n in each polyline and parameter values used to detect characteristic points.

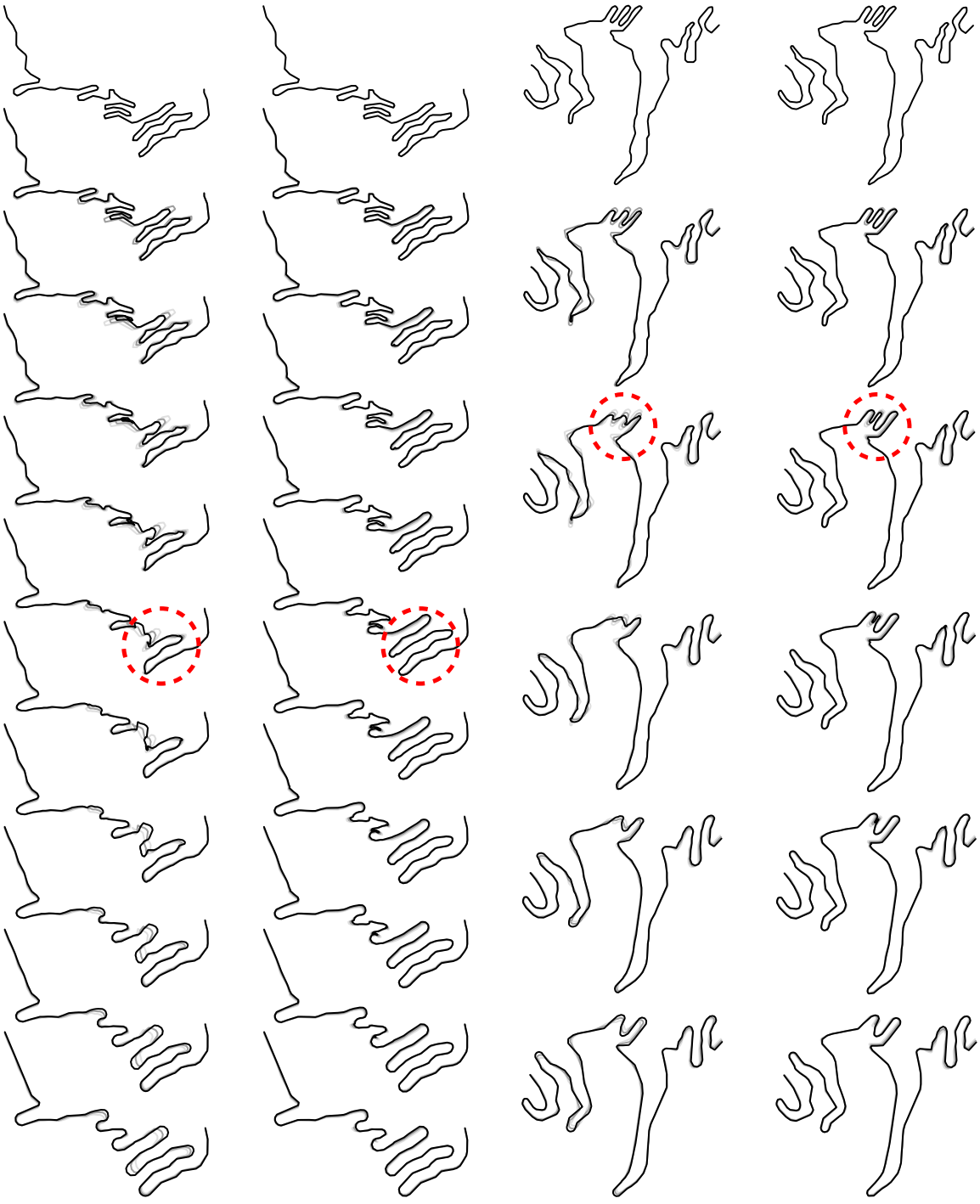
points that Algorithm 1 detected are marked by little squares. The parameter values used to obtain these results are listed in Table 1. Currently, these parameters must be set by trial and error, starting from base values of zero for $\sigma, \theta_i, \theta_b$ and θ_c and manually increasing them until the number and placement of characteristic points is acceptable to the user.

A sequence of snapshots[1] of the final morphs, after applying Algorithm 2, are shown in Figures 2(b) and 2(d), for Road 1 and Road 2 respectively. A look-back parameter K of 5 was used. For the purpose of comparison, Figures 2(a) and 2(c) show a simple linear morphing between the same polylines, where both polylines were uniformly parameterized to establish the correspondence between points. On a 3.0GHz Pentium 4 with 1GB RAM, the entire processing time was under 3 seconds for each example.

The optimum-correspondence morphing shows some clear improvements over the naïve linear morphing. The linear morphing in Figure 2(a) shows one of the large serpentine sections on the right being flipped "inside-out" during the morph. In contrast, the optimum matching morphing in Figure 2(b) simply expands the bends. It is evident that the total movement overall is much higher for the linear morphing than for the optimum matching morphing.

A similar situation occurs in the Road 2 example; the linear morphing collapses two bends to become one, where the optimum matching morphing simply collapses one of the bends and expands another to become the final bend.

[1]The full animations are available at http://i11www.iti.uni-karlsruhe.de/morphingmovies



(a) Road 1 (Linear) (b) Road 1 (OPTCOR) (c) Road 2 (Linear) (d) Road 2 (OPTCOR)

Figure 2: A comparison between simple linear morphing and the optimum-correspondence morphing (OPTCOR) for the two example roads. In each snapshot, the previous two frames are drawn in successively lighter shades of grey. Areas of particular interest are marked with dashed circles.

5 Concluding remarks

A number of improvements could be made to the algorithms in this paper with further research. Ensuring that self-intersections do not occur during a morph could potentially be accomplished by utilizing the algorithm of Surazhsky and Gotsman (2001b) to compute non-linear trajectories for points. Additionally, the detection of appropriate characteristic points with little or no user interaction requires further investigation.

6 Acknowledgements

The authors would like to thank Sébastien Mustière for providing the Carto2001 data. Martin Nöllenburg and Marc Benkert are supported by grant WO 758/4-2 of the German Research Foundation (DFG). National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

Alt, H. and Godau, M. (1995). Computing the Fréchet distance between two polygonal curves. *Int. J. of Computational Geometry and Applications*, **5**(1–2), 75–91.

Bespamyatnikh, S. (2002). An optimal morphing between polylines. *Int. J. of Computational Geometry & Applications*, **12**(3), 217–228.

Cecconi, A. and Galanda, M. (2002). Adaptive zooming in web cartography. *Computer Graphics Forum*, **21**(4), 787–799.

Cohen, S., Elber, G., and Bar-Yehuda, R. (1997). Matching of freeform curves. *Computer-Aided Design*, **29**(5), 369–378.

Efrat, A., Har-Peled, S., Guibas, L. J., and Murali, T. M. (2001). Morphing between polylines. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms (SODA'01)*, pages 680–689.

Erten, C., Kobourov, S. G., and Pitta, C. (2004). Intersection-free morphing of planar graphs. In *Proc. 11th Intern. Symp. Graph Drawing (GD'03)*, volume 2912 of *Lecture Notes in Computer Science*, pages 320–331. Springer Verlag.

Gomes, J., Darsa, L., Costa, B., and Velho, L. (1999). *Warping and Morphing of Graphical Objects*. Morgan Kaufmann.

Jones, C. B. and Ware, J. M. (2005). Map generalization in the web age. *International Journal of Geographical Information Science*, **19**(8–9), 859–870.

Lecordix, F., Jahard, Y., Lemarié, C., and Hauboin, E. (2005). The end of carto 2001 project: Top100 based on bdcarto database. In *Proc. 8th ICA Workshop on Generalisation and Multiple Representation*, A Coruña, Spain.

Lowe, D. (1989). Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, **3**(2), 119–130.

Misue, K., Eades, P., Lai, W., and Sugiyama, K. (1995). Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, **6**(2), 183–210.

Plazanet, C., Affholder, J.-G., and Fritsch, E. (1995). The importance of geometric modeling in linear feature generalization. *Cartography and Geographic Information Systems*, **22**(4), 291–305.

Samoilov, T. and Elber, G. (1998). Self-intersection elimination in metamorphosis of two-dimensional curves. *The Visual Computer*, **14**, 415–428.

Sester, M. and Brenner, C. (2004). Continuous generalization for visualization on small mobile devices. In P. Fisher, editor, *Developments in Spatial Data Handling – 11th Int. Symp. on Spatial Data Handling (SDH'04)*, pages 355–368. Springer Verlag.

Surazhsky, V. and Gotsman, C. (2001a). Controllable morphing of compatible planar triangulations. *ACM Transactions on Graphics*, **20**(4), 1–21.

Surazhsky, V. and Gotsman, C. (2001b). Morphing stick figures using optimized compatible triangulations. In *Proc. Ninth Pacific Conference on Computer Graphics and Applications (PG'01)*, pages 40–49.

van Kreveld, M. (2001). Smooth generalization for continuous zooming. In *Proc. 20th International Cartographic Conference (ICC'01)*, pages 2180–2185.

Weibel, R. and Dutton, G. (1999). Generalising spatial data and dealing with multiple representations. In P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, editors, *Geographical Information Systems – Principles and Technical Issues*, volume 1, chapter 10, pages 125–155. John Wiley & Sons.

Biographies

Damian Merrick has been a PhD student with the University of Sydney and National ICT Australia since March 2004. His research interests include graph drawing and computational geometry, with a focus on automatic generation of schematic diagrams and maps.

Martin Nöllenburg is a PhD student at Karlsruhe University since October 2005. His research interests include computational geometry and geometric networks, in particular their visualization such as generating schematic layouts of public-transport networks.