Eurographics/ IEEE-VGTC Symposium on Visualization 2008
A. Vilanova, A. Telea, G. Scheuermann, and T. Möller
(Guest Editors)

Volume 27 (2008), Number 3

COPERNICUS: Context-Preserving Engine for Route Navigation with Interactive User-modifiable Scaling

Hartmut Ziegler[†] Daniel A. Keim

University of Konstanz, Germany

Abstract

In this paper, we present an automated system for generating context-preserving route maps that depict navigation routes as a path between nodes and edges inside a topographic network. Our application identifies relevant context information to support navigation and orientation, and generates customizable route maps according to design principles that communicate all relevant context information clearly visible on one single page. Interactive scaling allows seamless transition between the original undistorted map and our new map design, and supports user-specified scaling of regions of interest to create personalized driving directions according to the drivers needs.

Categories and Subject Descriptors (according to ACM CCS): I.3.2 C.2.1 [Computer Graphics]: Graphics Systems I.3.3 [Computer Graphics]: Line and Curve Generation I.3.8 [Computer Graphics]: Applications

1. Introduction

Route mapping systems have emerged to one of the most popular applications on the World Wide Web. In February 2007, the keyword 'Routenplaner' (German for 'driving direction planner') has been the most popular search query on the German Google Zeitgeist [GZ07]. Despite the high demand, most of the systems on the web have not shown any significant new visualization techniques in previous years. Most of them do not distinguish between relevant and extraneous information, and interaction is usually limited to zooming and panning. We present COPERNICUS, a system for automated rendering of driving directions based on a topographic network graph metaphor with interactive scaling. We describe techniques for identifying, extracting and weighting relevant context information to generate a completely context-driven route mapping style, and introduce a novel scaling engine that allows a seamless interactive scaling between undistorted maps and our new map design.

When analyzing current route mapping systems, almost all systems today depict a specified region of the map and plot the route using a highlighting technique (see figure 1, we use this route as reference throughout the paper). As correctly identified before [Agr01], these types of maps have two main shortcomings: on the one hand, the constant scale factor leads to the problem that small roads near origin and destination are not visible, which requires zooming into these regions at different levels and to create multiple printouts. On the other hand, they do not distinguish between important and extraneous information, and clutter the map with useless information hundreds of miles away from the route.

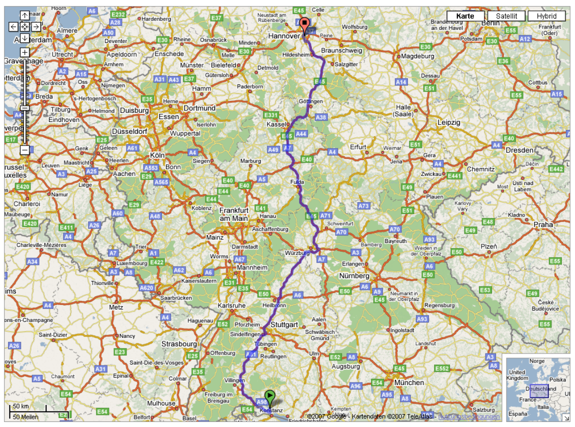

Figure 1: *With current highlighting techniques, short streets near origin or destination are not visible and require multiple zooming and panning steps as well as multiple printouts.*

[†] e-mail: hz@hzmail.de

2. Related Work and Definition of Design Goals

One very innovative system that appeared in 2001 and which tried to address these problems was LineDrive^{TM} [Agr01] [AS01], which focuses on generalization techniques that human map-makers use in hand-drawn maps and adapts them into an automated system. Figure 2 illustrates the same route as in figure 1 using LineDrive, which consists of a series of streets and turns. Similar to a verbal route direction, it focuses on the essential information for following a set of streets: street names, distances and turning points. As this technique adapts human generalization techniques like distortion and abstraction, each street has its distorted length, is clearly visible and conveys all information on a single page.

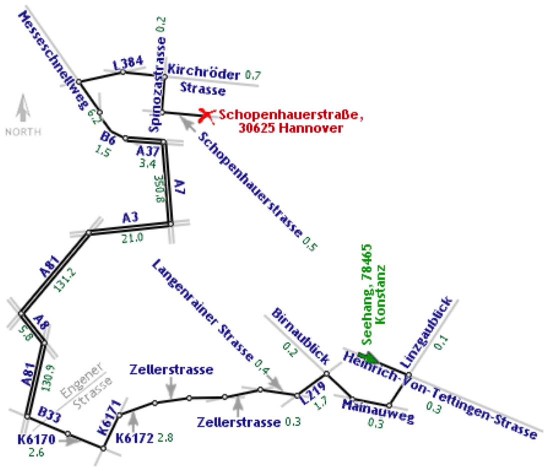

Figure 2: *The same route from Figure 1 using LineDrive which depicts a route by a sequence of streets with street names, lengths and turning directions. This works well in the USA where routes are mainly described by numbers, while Europeans often navigate by town names on traffic signs.*

Stripmaps or Overview/Focus maps try to solve the problem by using multiple maps with different scale factors on one page. However, this requires to puzzle the pieces together again and to determine which focus map belongs to which region of the overview map. Traditional techniques like fish-eye views cause regions outside the focus to be unreadable. An overview of cartographic generalization techniques can be found in the work of Agrawala [Agr01]. Further related work regarding the extraction and distortion process analyzes the schematization of networks [CBK*05] [KRB*05], wayfinding in hierarchical street networks [CF93] [TVP92], and personalized routes [PCS*06].

As pointed out in figure 1, a sophisticated route mapping system should be able to separate relevant from extraneous information, and focus only on relevant information. All information should be clearly visible on one single page, avoiding batches of printouts with different zoom levels.

LineDrive was an initial approach in this direction, but came with several shortcomings, especially for European citizens as driving directions are different to the number based system in the USA. Most importantly, relevant context information for navigation is missing. For example, no town names are communicated, so the driver cannot use traffic signs for orientation or navigation in order to verify if he is still on the correct route, or, even more problematic, at no position on the map does the driver know in which city he currently is (see figure 2). LineDrive also comprises a fault tolerance problem that once a driver misses a turn, it is nearly impossible to get back onto the correct route as the missing context information makes traffic signs useless to him.

It can be derived that route map visualization based only on street information is not sufficient to solve these problems. In fact, human driving directions are not only based on street names and numbers, but also on town names that have to be passed in order to reach a destination, therefore a map can also be seen as a schematized network with nodes and edges like in figure 3. A network graph approach that is automatically able to determine towns as nodes and its connecting streets as edges, and that is able to support orientation and navigation by identification and extraction of important context information, would be an promising solution. Furthermore, a seamless transition from an original map to our new map design would be beneficial. Instead of confronting the user with a maximum distorted map, this allows the user to interactively understand the distortion process by altering the map step by step. Figure 4 shows the stages of our prototype [Z06] which are described in the following section.

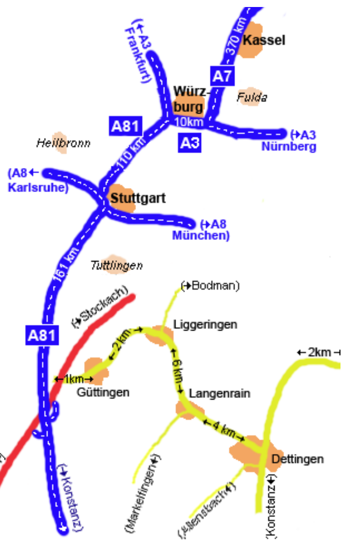

Figure 3: *A hand-drawn sketch of a street network according to our design principles. It allows navigation from node to node by using town names on traffic signs, but also shows cities left and right of the path to support orientation.*

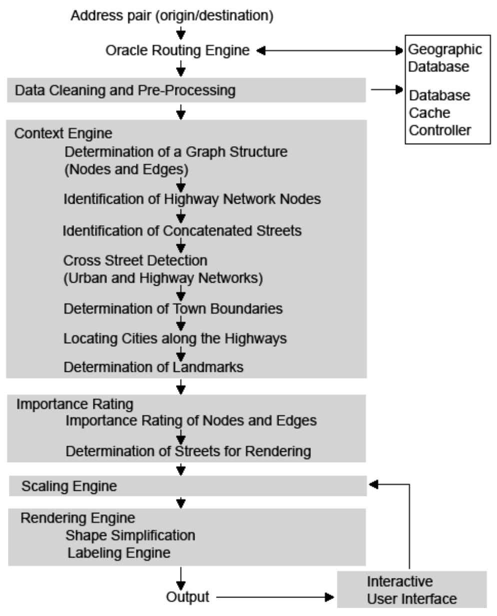
Address pair (origin/destination)
Oracle Routing Engine
Geographic Database
Database Cache Controller
Data Cleaning and Pre-Processing
Context Engine
Determination of a Graph Structure (Nodes and Edges)
Identification of Highway Network Nodes
Identification of Concatenated Streets
Cross Street Detection (Urban and Highway Networks)
Determination of Town Boundaries
Locating Cities along the Highways
Determination of Landmarks
Importance Rating
Importance Rating of Nodes and Edges
Determination of Streets for Rendering
Scaling Engine
Rendering Engine
Shape Simplification
Labeling Engine
Output
Interactive User Interface

Figure 4: *An overview of the system stages. After the context engine extracted the important information, heuristics rate their importance and build a hierarchy that is used by the scaling and rendering engine for generating our maps.*

3. The Context Engine

For generating our maps, extracting useful context information from the database is essential. On the one hand, this requires determination of what can be considered useful, on the other hand how algorithms can detect and extract this information from a database. Due to our fully modular concept, additional methods can easily be added.

3.1. Determination of Nodes and Edges

As stated in the design goals in section 2, the fundamental concept of our maps is based on a network graph metaphor that reflects towns as nodes and streets in between as edges, which allows navigation from node to node rather than from street to street compared to LineDrive. By analyzing the database which street belongs to which administrative area, we are able to detect towns and its connecting streets and can build the corresponding node/edge data structure.

3.2. Identification of Highway Network Nodes

Highway network nodes can be identified in the database by its data attributes. Although they are technically streets and therefore edges, we treat them as nodes because they represent important decision points in the distortion process. It is crucial at the highway network nodes that the map clearly reflects which ramp to take. In order to achieve this, we do not only extract the highways that a driver has to take, but also all other ramps that the driver has to avoid, and display them with light blue arrows along with the highway number and the destinations where he should not go to (see figure 6).

3.3. Identification of Concatenated Streets

The routing engine returns around 4500 street segments for our example route. 1800 of these segments are unique, all other carry the information redundantly with different street names or numbers (see figure 9). In order to determine connected streets segments that have the same name or number, these segments must be re-aligned. An algorithm processes all segments, and queues them in special data structures that now describe the streets. An importance rating determines which of them are used for rendering.

3.4. Cross Street Detection

Not every cross street is of relevance. If traveling a main street, small cross streets are often of no interest, and only intersections with other main streets are important. Vice versa, on a small street a small cross street might be of interest. The system compares the street types (1=highway, 2=country road, 3=main street, 4=road, 5=alley) which span a two-dimensional matrix that reflects the importance level of a cross street type compared to the current street type.

3.5. Determination of Town Boundaries

In order to display a region of the map as town if we travel inside its boundaries, distorting the exact geographic boundary is difficult. We therefore decided to symbolize towns with convex hulls (see figure 5) which retains the drivers attention on the relevant information of the map. The town boundaries also give the driver an exact feedback of his current location when entering and leaving a town.

3.6. Locating Major Cities along the Highways

On highways, it is often helpful to know the names of the major cities that are located along the route, even if they are not directly passed. Names of large towns and cities are widely visible on traffic signs, and in combination with distance measures give valuable feedback about the drivers current position on the map. The context engine identifies major cities in the vicinity of the highways with a spatial database query, plots its geometric shape onto the map, and helps the driver to build a topographic view of the map (see figure 6).

3.7. Determination of Landmarks

A set of rules evaluates if a landmark is shown (for example gasoline stations are often important and prominent landmarks), and a numeric threshold defines at which zooming level a landmark should start to be displayed. So far, we only added gasoline stations as landmarks, with other landmarks easily being added if required (see figure 7).

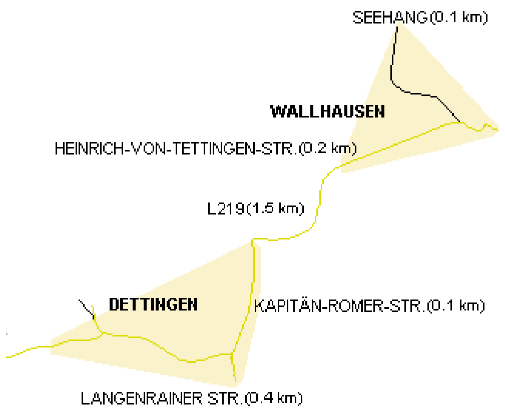
SEEHANG (0.1 km)
WALLHAUSEN
HEINRICH-VON-TETTINGEN-STR. (0.2 km)
L219 (1.5 km)
DETTINGEN
KAPITÄN-ROMER-STR. (0.1 km)
LANGENRAINER STR. (0.4 km)

Figure 5: *In order to display the nodes, we use convex hulls instead of the exact geographic shape of the town boundaries as they symbolize the nodes in a space efficient and easy to understand way. The driver can navigate large parts of the map by following the traffic signs from town to town, without the necessity to rely only on street names. The curved shape of the roads also gives additional feedback to the driver.*

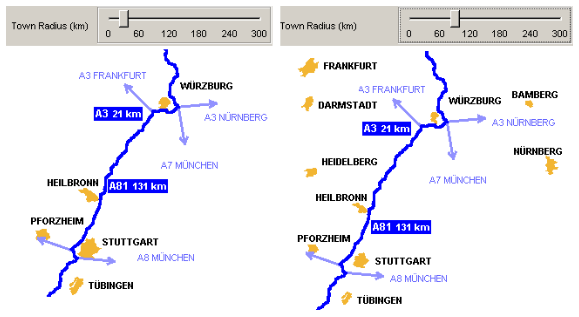

Figure 6: *Showing the major cities in the vicinity of the route facilitates orientation and navigation on the highway network, as the traffic signs give a permanent feedback to the driver. In our system, the user can use a slider to specify the radius in which the cities along the route are displayed.*

4. Importance Rating

After the context information has been extracted from the database, the next step analyzes the information and ranks its importance. It processes all information from the previous stage, and has the ability to accomplish substantial changes in order to improve the route map layout.

4.1. Importance Rating and Space Optimization

In this stage, a set of heuristics evaluates the importance of nodes and edges in order to optimize the network graph structure. Due to space limitations, we only describe one example of such a heuristic in order to demonstrate the principle: In country areas, it is often the case that villages or small

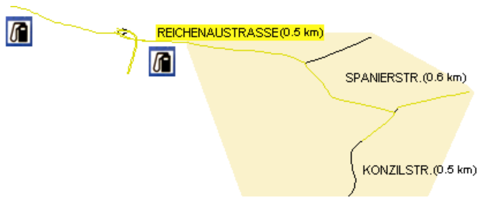

Figure 7: *A set of heuristics determines whether a landmark is important or not. So far, only gasoline stations have been added as landmarks, but other types can easily be added.*

cities are passed nearly on a direct way, with no important driving maneuver taking place. As each village is treated as one node with two edges, this consumes a lot of valuable display space. Heuristics try to recognize unimportant nodes and edges, and modify the data structure in order to reduce unimportant information (see example in figure 8). Nodes that are tagged unimportant are not required to be shown in detail and are removed from our data structure. In this case it is sufficient to represent the unimportant village just with a small symbol located along the way. Technically, 'two edges and one unimportant node' are replaced by 'one edge + one additional context information' in the data structure. This reduces the amount of streets to be rendered significantly.

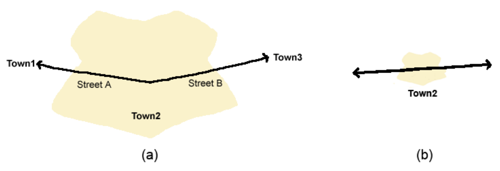

Figure 8: *One example of an importance rating heuristic: If a village is passed, and the combined length of streets inside the village is less than 2 miles, the number of streets inside the village is 3 or less, and the maximum turning angle from street to street is less than 20 degrees, then the small village is passed almost straight through. The node (a) is tagged 'unimportant' and replaced by a simplified view (b).*

4.2. Determination of Streets for Rendering

After evaluating the importance of nodes and edges, as a next step the system determines the streets that are used for being displayed and rendered on the map. As can be seen in figure 9 and as already mentioned in section 3.3, some street segments have several street names and street ID numbers, with different total street length. In order to determine the streets for rendering, the algorithm evaluates the importance of nodes and edges and the overlapping streets (street ID number or street name) together with their length to find a solution with as few streets as possible. All streets in important nodes must be shown in full detail and by their street name, whereas small streets in unimportant nodes are left out and two or more of those can be concatenated to one.

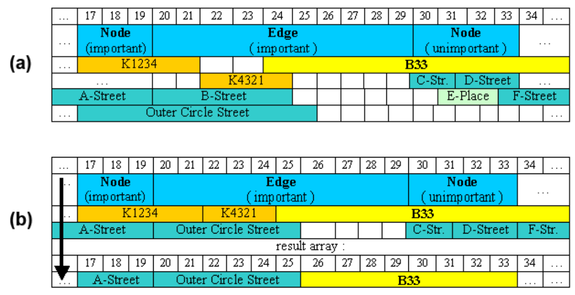

Figure 9: *After evaluating the importance of nodes and edges, an algorithm determines which street segments are suitable for rendering the maps. The decision depends on the importance of a node or edge, length of the street and street type (from 1 to 5, see section 3.4). The bottom line represents the streets that are finally used by the rendering engine.*

4.3. Determination of Street Lengths

In our system, the length of a street that is displayed on the map depends completely on the amount of important context information that has been determined for this street. Figure 10 gives an overview of some combinations of towns or important cross streets (see section 3.4) that might be correlated with a street. As an example, a cross street and a town (see figure 10 E) will require two extra space units for displaying the information on the screen in a clear looking way, but if the cross street is located inside the town (see figure 10 F), only one extra space unit will be sufficient as both are mapped one above the other. A space-saving algorithm computes the smallest amount of display space for each combination in order to use as few display space as possible. This value is finally the amount of display space that a street segment requests from the scaling engine in order to have enough space to show all its context information.

5. The Interactive Scaling Engine

Once the relevant context information has been retrieved and the data structures for nodes, edges and streets are determined, the scaling engine is able to construct a layout depending on the space that each street requests. As user-customizable maps can significantly facilitate the use of a route mapping system, our scaling engines allows seamless transitions between the original and our new map design.

5.1. Calculating an Initial Layout

Finding the optimal layout that takes all our design principles into account is an optimization problem, and it cannot be guaranteed that a map with the desired layout exists. For example, there are cases where the available display space is too small to show all context information. As a result, it is necessary to have an approximation process to come as close to the desired layout as possible. In contrast to LineDrive, our scaling engine is completely based on horizontal and vertical distortion of street segments, where the beginning of each street segment is attached to the end of the previous street segment. This construction can be larger or smaller than the viewport, so it is automatically scaled to fit the viewport to use all available space. As each street segment has an original length in the undistorted map, and a requested length depending on the amount of context information in our map (see figure 10), for finding the initial layout the algorithm iterates over all street segments. At each iteration the horizontal and vertical scaling for a street is increased by 20% or decreased by -20%. The multiplication is performed on both horizontal and vertical direction at the same time, so the geographic direction and shape of a street remains unchanged and only the size is altered. Such a change in the scaling is only permitted if consistency constraints are not violated (we will describe these in section 5.3). In case of a violation, the street element is not changed, and the iteration continues with the next street element until our final map layout is found. The final result of such a route map according to the design principles can be seen in figure 11.

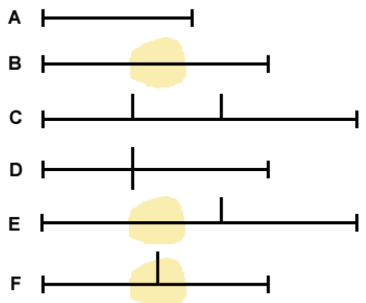

Figure 10: *The length of the streets to be rendered depends on the amount of context information that has to be displayed. Cross streets that leave left and right within a threshold are considered intersections and only need one space unit (D). A space-saving algorithm computes the minimum space that is required for each combination.*

5.2. Interactive Scaling and Distortion

Seamless scaling between original and distorted shape is a combination of several linear scale factors. In the undistorted state, the map preserves the aspect ratio and uses either the full height or full width of the viewport, depending on its shape (see figure 12 a). The maps generated by our route mapping system use the full height and width of the available display space. The distortion ratio can be adjusted by a slider between zero and hundred percent

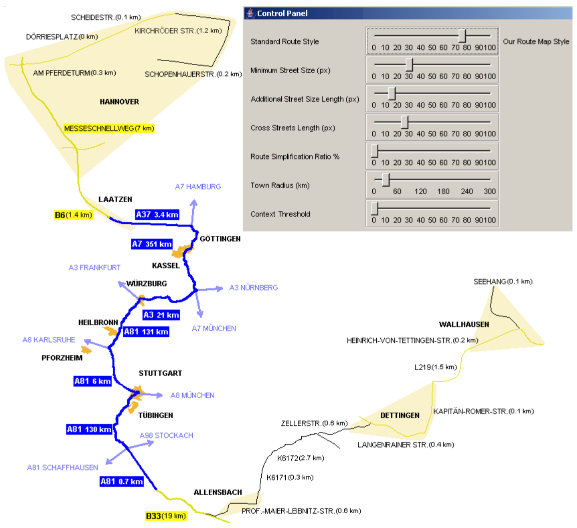

Figure 11: *This image shows the final result of a route map created by our system. In contrast to existing mapping styles, the network graph metaphor allows navigation by 'hopping' from town to town. The individual scale factor for each street allows to see all streets clearly visible on a single page, so only one printout is required.*

(1) Firstly, each street in the undistorted map (if we draw a bounding box around it) has a height of H_{undist} and a width of W_{undist}, each corresponding street in our distorted route map a height of H_{dist} and a width of W_{dist}. The scaling function is a linear transition in both horizontal and vertical direction, so the height of the street elements and bounding boxes varies linearly between H_{undist} and H_{dist} and its width between W_{undist} and W_{dist} depending on the distortion ratio.

(2) Secondly, due to the aspect ratio, the undistorted map uses either the full height or the full width of the viewport. In the example throughout this paper, the undistorted map uses the full height of the viewport $Hmax_{Viewport}$, but not the full width. Accordingly, the second scaling factor is a transition that linearly scales the width of our example map to the full width of the viewport $Wmax_{Viewport}$.

(3) Thirdly, each street is seamlessly scaled between its original length and a value where each context information for each street on the whole map has the same amount of space. This causes enlargement of short and shrinking of long streets. Therefore, streets with a lot of context information are larger than streets with fewer context information. However, each street has a defined minimal length to be clearly visible.

5.3. Consistency Constraints

As already identified in previous work [AS01], distortion of route maps can cause undesirable effects such as false or missing intersections, or inconsistent turn directions that have to be avoided.

Regarding false intersections that occur when intersections are created during distortion where they are not supposed to, a comparison of each of the 1800 street segments with all other street elements of our example would create enormous computational overhead ($n*(n-1)/2$ complexity). In order to accelerate the process, our engine uses minimum bounding rectangles to speed up the process (see figure 13). If the minimum bounding rectangles do not overlap, the streets within these rectangles will not overlap as well. This only requires a comparison of eight variables (x, y and $xsize, ysize$ for each pair of rectangles) in order to exclude these rectangles from a detailed consistency analysis.

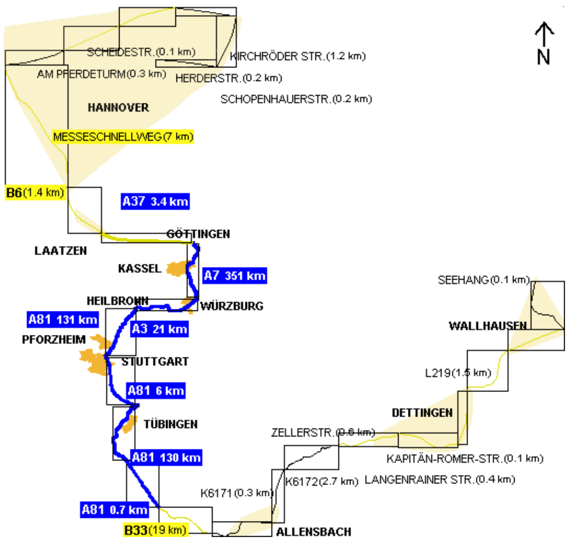

Figure 13: *The computation speed if false intersections occurred by the distortion algorithm can be significantly improved by using minimum bounding rectangles for each street. If two bounding rectangles do not overlap, there can also be no intersection of the streets that are enclosed.*

Regarding intersections that already exist in the original route, it is necessary to assure that these intersections are preserved and placed on the correct position during the distortion process. Figure 14 (a) shows a typical situation with three roads that form an intersection (the previous road is crossed via a bridge or tunnel, or a set of one-way roads require this maneuver). The distortion process might cause the intersection to vanish, or to be at the wrong position. Figure 14 (b) shows a solution where the three roads are handled as two roads, in this case it does not matter how the rectangles are distorted in horizontal or vertical direction as the intersection inside the large rectangle will always be at the correct position. An even better solution to redefine the street segments is to use the intersection point in order to create three separate rectangles as can be seen in figure 14 (c), in

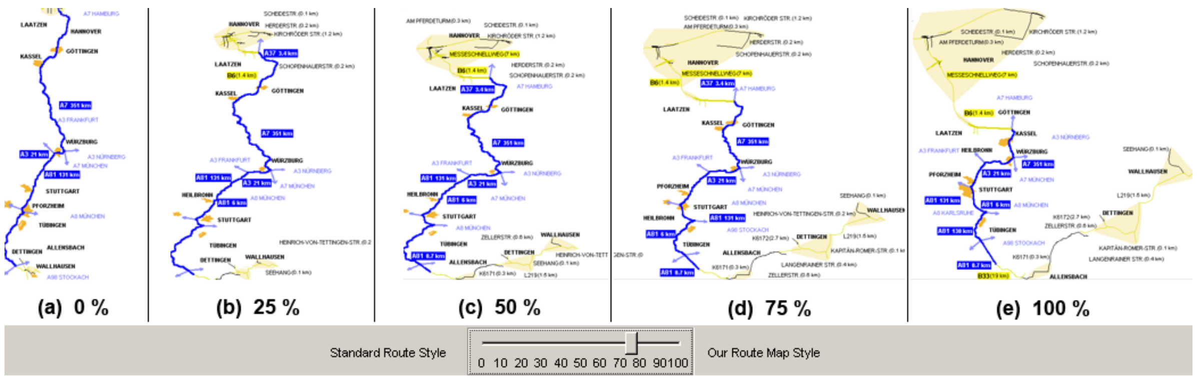

Figure 12: *The scaling engine allows a seamless interactive scaling from the original route map to our route map design. The system performs a seamless transition into a context-based network-oriented layout where all streets are clearly visible, and outlines the boundaries of towns in order to allow a navigation from node to node by following the road signs with town names.*

this case each of the three rectangles can also be distorted independently and the intersection point is still at the correct position.

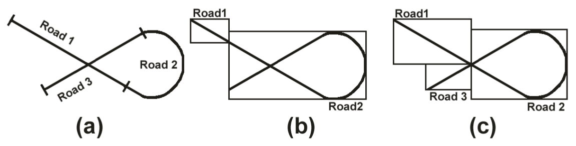

Figure 14: *Preserving intersections at the correct positions during the distortion process can be difficult. We solve this by redefining the affinity of street segments. As a result, each of the bounding rectangles can be distorted independently with the intersection remaining at the correct position.*

To maintain roughly the correct geographic direction of the streets and the turning angles between them, we exploit a side effect of the horizontal and vertical scaling. A street (see figure 15), will always go into roughly the correct geographic direction regardless of the distortion ratio, with a deviation of usually less than 25 degrees.

5.4. Shape Simplification

Shape simplification can be helpful to generate a clear map design, but in practical use can also be a big disadvantage because useful context information is lost that can give the driver valuable feedback while traveling. This is, for example, the case when the driver follows the path of a street with several curves and the curves can also be recognized on the map, giving him a clear impression of his current position, whereas a straight line that replaces the curves would leave him with no environmental feedback at all. Because a curve instead of a straight line is usually only consuming minimal additional display space, we prefer the unsimplified route shape, as the feedback to the driver is rather a benefit than a disturbing factor. The only road type suitable for full simplification might be highways.The rendering engine allows the simplification ratio to be adjusted by a slider (see figure 16).

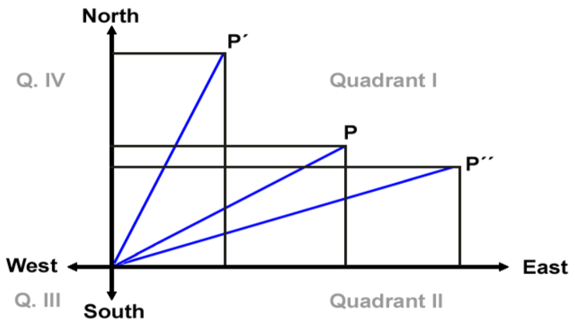

Figure 15: *One effect of linear scaling is that regardless of any vertical and horizontal distortion ratio a street will never be able to leave its quadrant. A street heading north-east remains between north and east, with extreme angles being unlikely so the general direction is maintained.*

6. Implementation Details

6.1. Data Sources and Implementation

In order to accomplish this project, we obtained geographic data of the German street network from NAVTEQ, and stored it in an Oracle 10g database with GIS extensions. We used Oracles "Network Data Model" to save the network graph with 5 million nodes (=intersections) and 10.6 million edges (=streets) in the database. Each edge has more than 100 additional attributes that our algorithms use to extract the contextual information. The NAVTEQ data also includes geographic data for town boundaries that we used for spatial queries, as well as landmarks. We also imported information from other databases such as population data to distin-

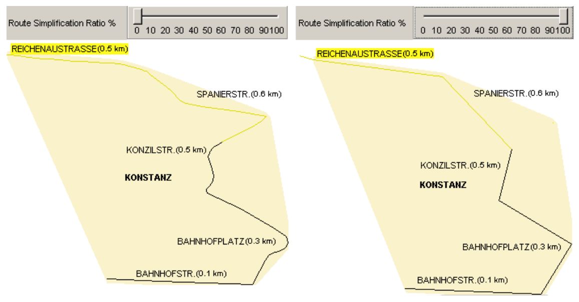

Figure 16: *The shape of a street can give the driver a valuable feedback if he is still driving on the correct road if the map reflects a characteristic curved path of a road. Our application allows to adjust the simplification rate individually.*

guish large cities from small ones. For calculating a path, we used the Oracle Routing Engine with 4GB memory cache, which created an extra 60GB of partitioning data on the hard disk, however, our application is independent from the underlying database system. For optimization purposes, all relevant context information is pre-computed and saved in additional street attributes, so a query retrieving the data for a street segment instantly returns all corresponding context information.This one-time pre-computation which requires to compare each street in the database with all context information takes around 120 hours, but offers instant retrieval of all context information afterwards. The query time for all 1800 segments is reduced from 40 to 2 seconds on a 2.4Ghz Pentium4, but can further be optimized. Database updates only require to update the corresponding tuples, and the system is fully scalable to larger networks. The client application is written in Java and therefore can be run in any web-browser.

6.2. The Rendering Engine

Once the scaling of the map has been computed, the rendering engine uses the data to render the output. The engine draws towns, cities and streets, places labels (for streets, towns, cross streets and where they lead to), distances, landmarks, north orientation arrow, and other context and decorative elements, priorized in this order. Context information with no available display space is not mapped so there is no overlap. Different types of streets are characterized by different line width and color, and can easily be adapted to local conditions (highways are colored blue in Germany, green in Switzerland, and red in many other countries). The control interface allows to interactively control the scaling and rendering engine, like manipulating the overall distortion ratio, varying the minimum street size or the length of the cross streets (see figure 10), shape simplification, or the radius of the vicinity in which major cities along the route shall be displayed. The application is not limited to these parameters, and more controls can easily be added.

7. Conclusion and Future Work

In this work, we presented a context-preserving route mapping prototype system with a novel scaling engine that renders route maps in a network graph metaphor which allows traveling a path from node to node by mainly using context information. The application evaluates and extracts a variety of important context information from a database, ranks their importance, and offers an advanced scaling engine that displays all relevant information clearly visible on one single page. The scaling engine supports a seamless interactive distortion between a familiar undistorted map and our context-based mapping style. Future work focuses on rendering a more attractive map design, a usability test, integration of more landmarks, linking landmarks directly to corresponding web pages, and optimizing the routing engine to calculate simpler to navigate paths [DK03] [PCS*06].

Acknowledgements: The authors thank Navteq for the geographic data, Holger Bruch (Logiball) for the GDF Suite Software, and Chuck Freiwald and Siva Rivada (Oracle) for their help with the Oracle routing engine. Patent pending.

References

[Agr01] Agrawala M.: *Visualizing Route Maps*. PhD thesis, Stanford University, 2001.

[AS01] Agrawala M., Stolte C.: *Rendering Effective Route Maps: Improving Usability through Generalization*. ACM Press, SIGGRAPH 2001, Computer Graphics Proceedings, pp. 241-250, 2001.

[CBK*05] Cabello S., de Berg M., van Kreveld M.: *Schematization of networks*. Computational Geometry: Theory and Applications,Vol.30 n.3 p.223-228, 2005.

[CF93] Car A., Frank A.U.: *Hierarchical Street Networks as a Conceptual Model for Efficient Way Finding*. Proc. of the EGIS'93, Italy, pp. 134-139, 1993.

[DK03] Duckham M., Kulik L.: *"Simplest" Paths: Automated Route Selection for Navigation*. COSIT'03, Lecture Notes in Computer Science, Springer, 2003.

[GZ07] google.com/press/zeitgeist/zeitgeist-feb07.html

[KRB*05] Klippel A., Richter K., Barkowsky T., Freksa C.: *The Cognitive Reality of Schematic Maps*. in Meng, Zipf, Reichenbacher (Eds), Map-Based Mobile Services-Theories, Methods and Implementations, 57-74

[PCS*06] Patel K., Chen M., Smith I., Landay J.: *Personalizing routes*. Proc. of the 19th ACM symposium on User interface software and technology, 2006.

[TVP92] Timpf S., Volta G.S., Pollock D.W.: *A Conceptual Model of Wayfinding Using Multiple Levels of Abstraction*. In Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Springer Lecture Notes, Vol 639, pp 348-367, 1992.

[Z06] Ziegler H.: *Context-Preserving Route Map Visualization*. Master Thesis, 2006.