Continuous Generalization for Small Mobile Displays

Claus Brenner and Monika Sester
Institute of Cartography and Geoinformatics, University of Hanover, Germany
{claus.brenner, monika.sester}@ikg.uni-hannover.de

Introduction

Visualization of spatial information on small displays is vital for several new emerging applications, notably location based services and car navigation systems. Typically, these applications have to rely on small mobile devices, that have a small display on the one hand, and also limited storage and computing capability on the other hand.

In order to communicate spatial information in both overview and detail, the system has to dispose of the possibility of flexibly zooming in and out. This requires that different levels of detail (LOD) are available and can be presented adequately. There are several approaches available in order to provide multiple levels of detail: most of them rely on pre-generalized representations, possibly stored in a multiple resolution data base (MRDB). Also, there are attempts for implementing real-time functionality using web-technology [Sarjakoski et al. 2002]. All of these approaches are based on well-known generalization functions, e.g. simplification, elimination, or aggregation.

Most of these operations are discrete in nature, leading to discrete changes in the representation of spatial objects. Typically, they are pre-computed for certain fixed scales so that when a certain scale is requested by a display device, the appropriate data base is selected and displayed. However, this leads to the well-known popping effects encountered when switching between two LOD's while continuously zooming in or out.

In order to reduce or eliminate these effects, we are aiming at a continuous generalization – comparable to Hoppe's approach of progressive meshes for the simplification of general triangulated surfaces [Hoppe 96]. This involves to represent cartographic objects in terms of their most simple geometry plus a sequence of elementary operations transforming them into the most detailed geometry available. This allows a client not only to decide which number of elementary operations to apply, but also fits nicely into a streaming concept which enables a server to successively improve the display quality on a client through a limited bandwidth channel. In this paper, this approach is exemplified with a specific simplification operation, namely the simplification of building ground plans.

Applications: Car and Personal Navigation Systems

Modern car navigation systems have been introduced in 1995 in upper class cars and are now available for practically any model. They are relatively complex and mature systems able to provide route guidance in form of digital maps, driving direction pictograms, and spoken language driving instructions. Looking back to the first beginnings in the early 1980s, many nontrivial problems have been solved such as absolute positioning, provision of huge navigable maps, fast routing and reliable route guidance.

Car navigation systems use map data acquired by map database vendors such as Tele Atlas or NavTech and supplied to car navigation manufacturers in an exchange format (e.g. GDF). There, it is converted to the proprietary formats finally found on the map CD or DVD. This conversion is highly nontrivial since the data has to be transformed from a descriptive form into a specialized form supporting efficient queries by the car navigation system. For example, all systems use a hierarchy of maps of increasing generalization level, spatial data structures, cross links and indices. Often, structures and values are pre-computed by this conversion process in order to relieve the navigation system's online resources such as bandwidth and CPU time. Thus, the pre-computation of generalization levels, as proposed in this paper, would fit nicely into the overall processing scheme.

In order to support a driver to recognize the current situation from a map, a possible extension of today's map displays would be if, besides the road network and points of interest, building ground plans would be integrated, either in a 2D map display or as 2.5D extruded buildings. However, looking at existing sources like cadastral maps, one finds that they contain highly detailed objects, which poses a problem for navigation systems regarding the absolute amount of storage required as well as bandwidth limitations during read and draw of the data. Thus, it is clear that map generalization should play a role in those systems.

Several types of car navigation systems are discussed today (see Figure 1). Traditional systems where all processing is done in the vehicle itself, using map data obtained from a local drive (typically CD or DVD, nowadays also HDD drives) are termed on-board systems. Their advantage is that they are not dependent on external information, except for GPS signals from time to time. The main disadvantage is that the on-board database usually is quite out-of-date since their distribution is in year- or half-year cycles and, even worse, most car owners are usually not buying the updates anyhow – so on average the on-board maps are several years old. This causes not only some inconvenience but can also lead to severe navigation errors since map matching algorithms are used internally for positioning.

Recently, off-board systems have been introduced into the market. Their particular characteristic is that route computations are done on a central server and the role of the on-board unit is restricted to positioning and route guidance. Their main advantage is that the server based route computation can potentially use more sophisticated algorithms, up-to-date maps, and even current traffic information. On the other hand, due to the communication necessary, those systems might react slower to user input or in case the driver makes a wrong manoeuvre.

Finally, hybrid navigation systems are discussed which are able to operate self-sufficiently but can get map updates, current traffic information, or additional map or tourist information from a server when requested.

From this discussion, it is clear that techniques which are able to provide map data on different generalization levels and in some progressive form, streaming them over narrow bandwidth channels, are especially suited for off-board and hybrid navigation systems. However, even on-board systems could use such an approach, since usually loading the map from CD or DVD in a car environment is quite bandwidth limited, a situation which will get worse when additional more detailed geometry such as ground plans from cadastral maps is to be displayed.

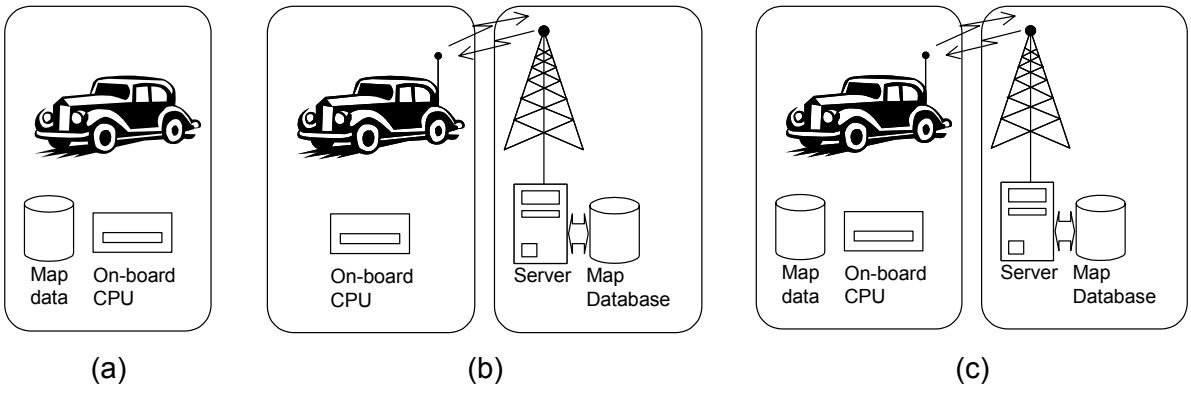

Figure 1: Basic types for car navigation systems: on-board (a), off-board (b), and hybrid (c).

Map Generalization

Generalization is a well known and important operation in cartography: in order to generate maps of different scales, a set of operations like simplification, aggregation, displacement or typification have to be applied. Whereas generalization can be achieved manually by human experts – since more than 30 years research has been conducted in order to automate this

process. The current situation shows many research concepts and proposals for algorithms for dedicated generalization operations, and even some products on the market. What is, however, not yet solved is a comprehensive solution that allows several operations to interplay, e.g. integration of simplification, typification and displacement. Communication of spatial information to small displays has put a new challenge to cartography and generalization, and already triggered a series of new applications.

In car navigation, there is a series of fixed scales, that are presented to the user as soon as he/she changes the zoom level. There are also applications for PDA's, that allow flexibly zooming in and out using a helicopter metaphor; also here, however, dedicated, pre-calculated scales are presented at predefined scale levels. The change between the different scale levels is typically "hidden" using blending or morphing techniques. Van Kreveld [2001] presents ideas for realization of such methods.

In this paper, we want to demonstrate, how the popping-effect can be eliminated when going from one discrete representation of an object to the other by adopting the progressive meshes approach from mesh simplification. The application is shown for the generalization of building ground plans.

There are several approaches for the generalization of building ground plans (e.g. [Staufenbiel 1973], [Mayer 1998], [Lamy et al. 1999], [Mackaness 2001]). In our approach [Sester 2000], we use a set of rules that are applied to each individual building façade. The only parameter for the algorithm is a minimal façade length that is just perceivable in a given representation or scale. Based on this threshold, each façade is inspected with respect to its length: if a façade length is smaller than the threshold, it has to be eliminated and replaced adequately. The fundamental control parameter of the approach is the minimal length of a building façade in the generalized representation, as façades that are shorter than this length cannot be perceived appropriately in the generalized situation. The simplified ground plan is determined by a local analysis of all façades that are below this threshold. Three cases have to be considered, depending on the direction of the predecessor and successor s_{n-1} and s_{n+1} of a short line s_n that has to be replaced (see Figure 2):

1. **Predecessor and successor have same direction:** this situation represents an offset: the longer edge is intersected with the next approximately orthogonal edge. In Figure 2a), the successor edge s_{n+1} is longer than the predecessor of s_n, thus it is intersected with edge s_{n-2} and edges s_{n-1} and s_n are removed. In a similar way, a longer predecessor edge would have to be intersected with s_{n+2}.
2. **Predecessor and successor have opposite direction:** this is an extrusion or intrusion: the extrusion is cut to the length of the shortest edge among predecessor or successor. In Figure 2b) s_{n-1} is shorter, thus s_{n-2} is intersected with s_{n+1} and s_n and s_{n-1} are eliminated completely.
3. **Other cases**: represent a corner: predecessor and successor edge are intersected to form a new building shape point, see Figure 2c).

All facades of a building are inspected iteratively using the mentioned rules, until all facades are above the given length. This also includes the ones that are generated within the process.

There are cases, where the mere analysis of a short facade is not sufficient: a long, narrow part of a building denotes a significant characteristic and thus cannot be simply cut off, even if the narrow side is below the given threshold. As a criterion for preservation, the size of the building part can be used. Consider Figure 2b): the size of the building part that is cut off by the operation is the product of s_{n-1} and s_n. If this size is above a given threshold (typically the square of the minimal facade length) the small facade is not eliminated, but enlarged, and thus the building part is emphasized.

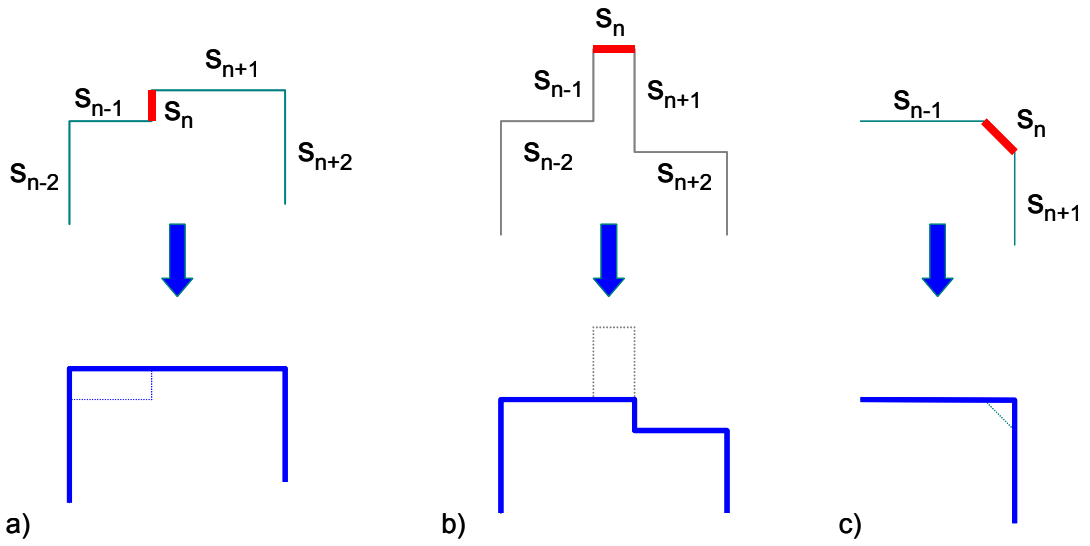

Figure 2: Elimination of short facade s_n: offset, intrusion/extrusion and corner.

In Sester [2001] the algorithm is extended by an adjustment process: the ground plans derived with the rule based approach is taken as approximate values from which a parametric representation of the building is derived, describing the object in terms of its width(s) and length(s). These parameters are refined using least squares adjustment. This has the effect, that resulting facades are “averaged” among the original facades, leading to an intermediate position. Also, the adjustment process allows for the possibility of enforcing certain properties of the object, e.g. emphasizing a small elongated building part.

Using the above mentioned algorithm, the following results can be obtained: Figure 3 shows an extract of ground plans of a rural village. The sequence visualizes the original situation on the left and two generalizations with different minimal facade widths. Clearly the increasing simplification can be observed.

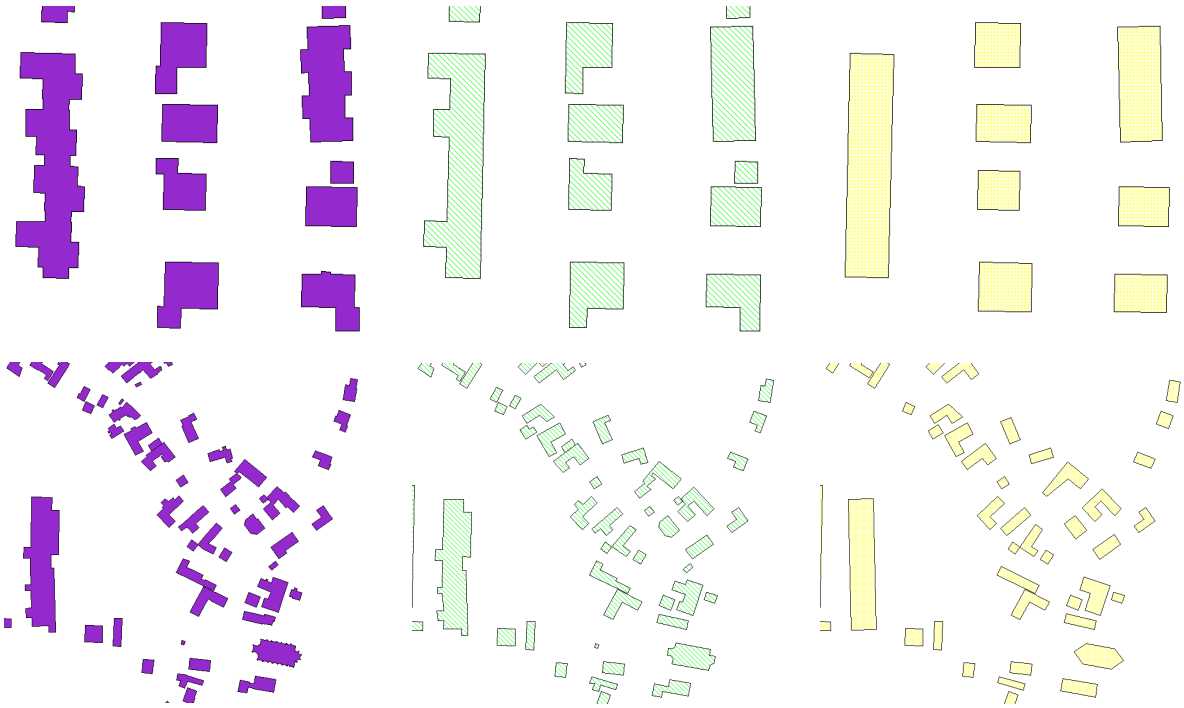

Figure 3: Building simplification – two examples: Original (left row), simplification with minimum facade width of 3m (middle), simplification with minimum facade width of 7m (right).

In dense city areas, typically, a merging of adjacent buildings has to be done first, in order to avoid overlapping of generalized versions of the buildings (see Figure 4). Figure 5 shows the result of a simplification followed by the aggregation.

Figure 4: Original (left), result (right).

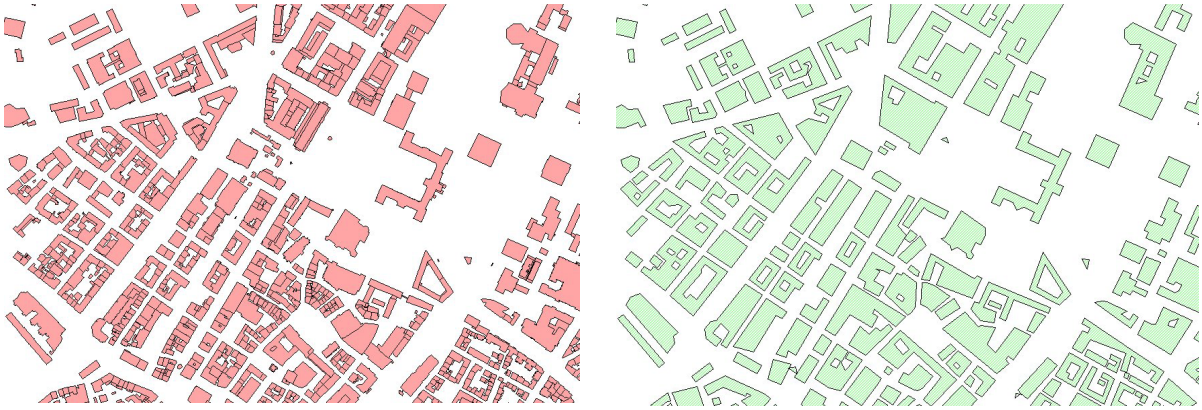

Figure 5: Original (left), result (right): after aggregation of adjacent objects and simplification.

Continuous Generalization

The Generalization Chain

Similar to the ideas introduced by Hoppe for triangulated meshes [Hoppe 96], we define for a polygon P consisting of n vertices a minimal representation P^m, with $m \leq n$ vertices, and a maximal representation $P^n \equiv P$, consisting of all original vertices. The minimal representation is the one which is still sensible from a cartographic viewpoint, for example a rectangle, $m = 4$ or the empty polygon.

During pre-processing, map generalization starts from polygon P^n, successively simplifying its representation using generalization operations as described above, finally yielding polygon P^m. Assume that k generalization steps are involved (each leading to one or more removed polygon vertices), and the number of polygon vertices are numbered $i_0 = n$, i_1, …, $i_k = m$, then a sequence of generalized polygons

$$P \equiv P^n \equiv P^{i_0} \xrightarrow[g_0]{} P^{i_1} \xrightarrow[g_1]{} \ldots \xrightarrow[g_{k-1}]{} P^{i_k} \equiv P^m$$

is obtained, where g_j denotes the j-th generalization operation. Every generalization step g_j is tied to a certain value of a control parameter ε_j, which relates to the display scale and can be – as discussed above – for example the length of the shortest edge in the polygon.

Thus, we can think of ε_j as the length of the edge which was eliminated during generalization step g_j or alternatively as the length of the shortest edge in polygon P^{i_j}. Since generalization proceeds using increasing edge lengths, the sequence of ε_j is monotonically increasing. As a first consequence of this, one could pre-compute and record all operations g_j, in order to derive quickly any desired generalization level ε by the execution of all generalization operations g_0, …, g_j, where $\varepsilon_0,...,\varepsilon_j \le \varepsilon$ and $\varepsilon_{j+1} > \varepsilon$.

However, it is obvious that for most applications, the inverse operations g_j^{-1} are more interesting, producing a more detailed polygon from a generalized one. Thus, we have the sequence

$$P^m \equiv P^{i_k} \xrightarrow[g_{k-1}^{-1}]{} P^{i_{k-1}} \xrightarrow[g_{k-2}^{-1}]{} \ldots \xrightarrow[g_0^{-1}]{} P^{i_0} \equiv P^n,$$

where again one can decide up to which point the polygon modification should be carried out, characterized by the corresponding parameter ε. This way, the inverse generalization chain can be used for progressively transmitting information over a limited bandwidth channel by transmitting P^m followed by a sufficient number of inverse generalization operations.

Encoding Elementary Generalization Operations

Some generalization operations have been introduced above. We can call them elementary generalization operations (EGO's), because every generalization chain will be made up of a combination of EGO's. Each EGO in turn consists of one or more simple operations (SO's) modifying the polygon. It is obvious that there are operations which modify the topology of a polygon, namely the insertion and removal of vertices, and operations which affect the geometry only. Table 1 shows a list of simple operations. This list is not minimal, since e.g. a "`DV i`" operation is equivalent to "`IV i,0`". However, for convenience and for achieving a most compact encoding, the operations might be defined redundantly. Knowing the parameters of a simple operation allows to immediately give the inverse operation except for the "remove vertex" operation for which the inverse would require an additional parameter to specify the location of the vertex to be inserted.

Opcode	Description	Parameters	Inverse Operation
`IV`	Insert Vertex	`IV` <edge id> <rel. position>	`RV` <edge id + 1>
`DV`	Duplicate Vertex	`DV` <vertex id>	`RV` <vertex id + 1>
`MV`	Move Vertex	`MV` <vertex id> <dx> <dy>	`MV` <vertex id> <-dx> <-dy>
`RV`	Remove Vertex	`RV` <vertex id>	–

Table 1: Simple operations used to define more complex EGO's.

Figure 6 shows how SO's combine to an inverse EGO. Starting from the left polygon consisting of a simple rectangle, a number of SO's is applied in order to obtain the more complex L-shaped polygon to the right. Note that infinitely many combinations of SO's can be used to obtain the same EGO. As long as a sequence does not contain remove vertex operations, it can be immediately reversed from a stored history of operations.

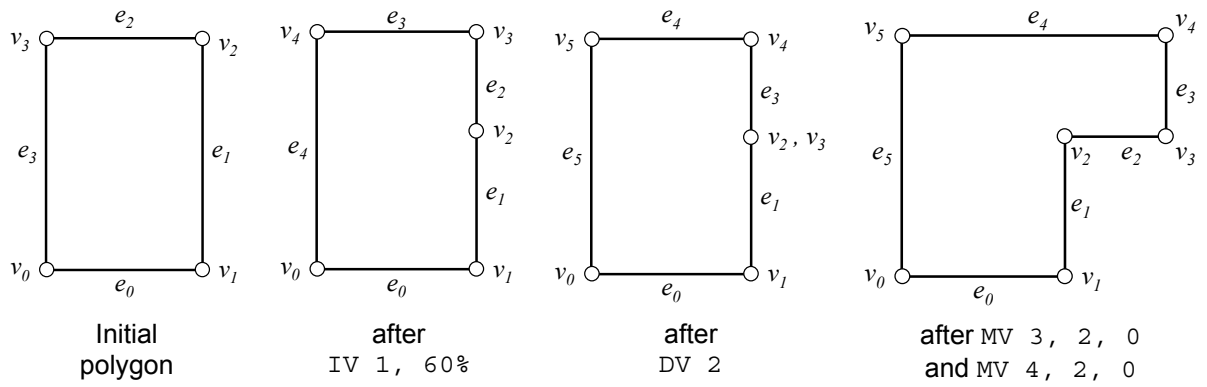

Figure 6: Example for an inverse EGO, forming an L-shaped building from a rectangular building. The EGO is decomposed into four SO's.

A Client-Server Communication Scheme for Progressively Streaming Map Data

To describe the mechanisms of a progressive streaming of map data, we now introduce the notion of a client and a server. In case of internet map displays, off-board car navigation as well as personal navigation systems, these take just the roles as expected. However, in other applications, they might be defined differently. For example, on-board car navigation systems might define the server as the main CPU unit where the mass storage resides, and the client as being the head unit CPU used for map display and user input.

One possible realization is depicted in Figure 7, based on the assumption that the server keeps track of the state of the client. A stateless approach could be used instead, however this would imply a larger amount of communication, telling the server each time the object id's and generalization levels present in the client in order to allow the server to compute the appropriate differential SO's.

When the user requests a new part of the map, the client is able to compute the bounding box in world coordinates and the generalisation level ε, the latter being based on the scale as well as some preferences which could balance speed versus "map quality". The client sends this information to the server which can retrieve the appropriate objects from the database. Since the server keeps track of which objects have already been sent to the client, it can deduce the appropriate SO's needed to update the display and send them to the client. While receiving SO's, the client will constantly refresh its display. If the user interacts before the entire set of SO's has been sent, the client may send a break request to the server which in turn will stop sending SO's. There might be additional communication items, for example to allow the client to drop objects currently out of view to conserve memory.

An Extension: Continuous Generalization

When a map representation is switched due to generalization, this usually leads to a visible "popping" effect. Compared to switching between different, fixed levels of detail, the use of EGO's is already an improvement, since it gradually modifies the polygon rather than just replacing it as a whole.

However, one can still improve on this. Intermediate states can be defined which continuously change the object in response to an EGO. For example, a "collapse extrusion" EGO (see Figure 2b) would be interpreted as "move extrusion until in coincides with the main part, then change the topology accordingly". We term this approach *continuous generalization* as it effectively allows to morph the object continuously from its coarsest to its finest representation.

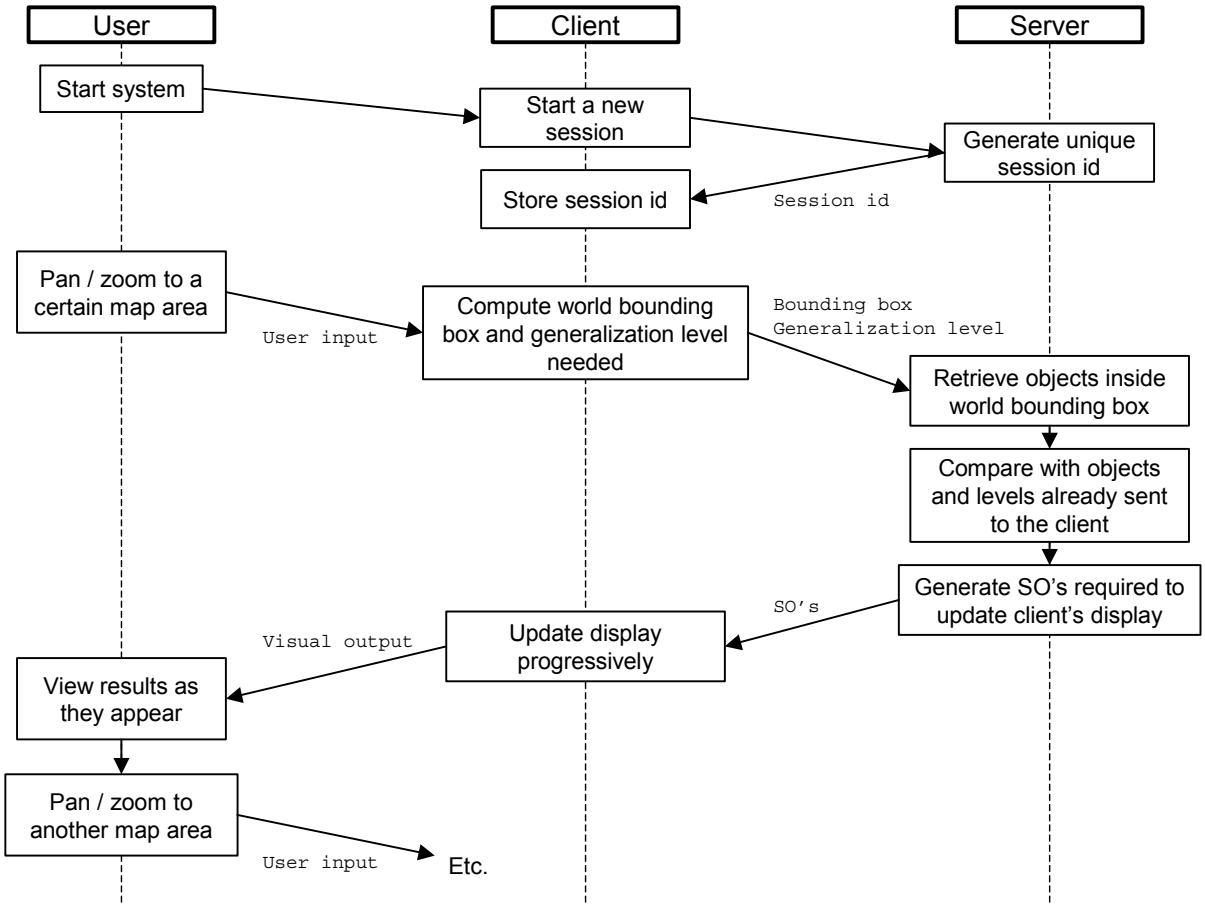

Figure 7: Example interaction diagram for client-server communication.

Since each EGO is made out of one or more SO's, their effects on display popping has to be taken into account. However, this is trivial, since we can deduce immediately that `IV` and `DV` are not changing the object's geometry. `RV` will only lead to a visible effect if the vertex, its predecessor and its successor are non-collinear. Thus, `MV` is the only remaining SO that has to be regarded. This means that a continuous generalization can be achieved by using an appropriate encoding of EGO's in terms of SO's, together with an animation in the client which gradually shifts vertices instead of moving them in one step upon encountering a `MV` operation.

Conclusions and Outlook

We have shown a method to incorporate standard generalization operations into a client/ server context, motivated by applications like car and personal navigation systems or in general applications using small displays and/or a limited bandwith database access. For this, elementary generalization operations (EGO) were defined, which in turn are composed of a common set of simple operations (SO). SO's can change the topology as well as the geometry of polygonal objects. This makes it possible to progressively transmit objects by sending an initial simple geometry and a sequence of SO's modifying this to obtain a detailed geometry. Furthermore, we have shown how a slight modification of this method can be used to obtain continuous generalization, where abrupt changes in the objects are avoided by using animated simple operations. Thus, the sequence of SO's effectively is a set of instructions describing how a continuous morph between different object representations has to take place.

For the future we plan to extend the concept to other cartographic generalization operations like displacement, which is straightforward, but also more complex operations like aggregation and typification.

References

Hoppe, H. [1996], Progressive Meshes. Proceedings of SIGGRAPH 96 (New Orleans, LA, August 4-9, 1996). In *Computer Graphics* Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 99 – 108.

Kreveld, M. van [2001], Smooth Generalization for Continuous Zooming, Proceedings of the ICC, Beijing, China, 2001.

Lamy, S., Ruas, A., Demazeau, Y., Jackson, M., Mackaness, W. & Weibel, R. [1999], The Application of Agents in Automated Map Generalization, in: Proceedings of the 19th International Cartographic Conference of the ICA', Ottawa, Canada, CD-Rom.

Mayer, H. [2000], Scale-Space Events for the Generalization of 3D-Building Data Adjustment, in: `International Archives of Photogrammetry and Remote Sensing', Amsterdam, Netherlands, Vol. XXXIII, Part B4, pp. 639-646.

Rainsford, D. & W. Mackaness [2001], Template Matching in Support of Generalization of Rural Buildings, in: Proceedings of the Joint International Symposium on "GeoSpatial Theory, Processing and Applications" (ISPRS/Commission IV/SDH2002), Ottawa, Canada, July 8-12, 2002, CD-ROM, (2002).

Sarjakoski, L.T Sarjakoski, L. Lehto, M. Sester, A. Illert, F. Nissen, R. Rystedt, and R. Ruotsalainen, Geospatial Info-mobility Services - a Challenge for National Mapping Agencies. Proceedings of the Joint International Symposium on "GeoSpatial Theory, Processing and Applications" (ISPRS/Commission IV/SDH2002), Ottawa, Canada, July 8-12, 2002, 5 p, CD-ROM, (2002).

Sester, M. [2000], Generalization Based on Least Squares Adjustment. In: International Archives of Photogrammetry and Remote Sensing, Amsterdam, Netherlands, Vol. XXXIII, Part B4, pp. 931-938.

Sester, M. [2001], Maßstabsabhängige Darstellungen in digitalen räumlichen Datenbeständen, Habilitationsschrift, Deutsche Geodätische Kommission, Reihe C, Heft 544.

Staufenbiel, W. [1973], Zur Automation der Generalisierung topographischer Karten mit besonderer Berücksichtigung großmaßstäbiger Gebäudedarstellungen, PhD thesis, Fachrichtung Vermessungswesen, Universität Hannover.