Continuous Generalization for Visualization on Small Mobile Devices

Monika Sester and Claus Brenner

Institute of Cartography and Geoinformatics, University of Hannover,
Appelstr. 9a, 30167 Hannover, Germany
monika.sester@ikg.uni-hannover.de, claus.brenner@ikg.uni-hannover.de

1 Introduction

Small mobile devices are becoming more and more important for the visualization of spatial information. This tendency is on the one hand driven by the availability of such small devices with displays (e.g. mobile phones, PDA's), as well as by a growing number of popular applications, namely Location Based Services (LBS) in general and navigation in particular.

However, these small devices also have limitations: first of all, the computing and storage capabilities are limited, leading to the situation that huge data sets or complicated calculations cannot be loaded or run on the devices, respectively. Furthermore, the displays are quite small and of relatively low resolution so that the visualization is not comparable to a map presentation.

In order to still be able to employ this new technology for mobile navigation, the following demands can be formulated: firstly, spatial data sets in different resolutions or levels of detail have to be made available for the mobile user in order to allow for a flexible zooming in and out for getting both detail and overview information. Secondly, this information has to be transmitted to the user from a server taking the possibly limited bandwidth of the communication channel into account.

In the paper a solution is presented that allows for a progressive transmission of spatial data to a small mobile device, starting with coarse overview information and progressively streaming more detailed information to the client. This information will be interpreted by the client and animated in a way to simulate a continuous generalization from coarse to fine and vice versa. A prototype has been developed to illustrate the functionality and evaluate the results.

The paper is structured as follows: after a brief review of related work, a set of elementary generalization operations (EGO's) is defined, which allows for the modification of spatial objects. Based on this elementary vocabulary, two generalization applications are described: generalization of building ground plans and typification of buildings. The last section describes the client-server

architecture, as well as an extension to gradually animate the changes in order make the discrete changes less visible. A summary and outlook concludes the paper.

2 Related Work

In order to provide adequate information to a user, it has to be adapted to the personal situation, the device used, the task to be solved, etc. (see e.g. [Nivala, Sarjakoski, Jakobsson & Kaasinen 2003, Reichenbacher 2004]). One major component is the adaption to the level of detail of the information. In cartography, the derivation of information in different levels of detail is being achieved using generalization operations [Hake, Grünreich & Meng 2002], which traditionally have been applied by human cartographers. Over 40 years of research in the automation of generalization processes have lead to a series of algorithms specified for dedicated purposes, e.g. line generalization [Douglas & Peucker 1973], area aggregation, building simplification, typification [Regnauld 1996, Sester 2004] or displacement [Højholt 1998, Harrie 1999, Sester 2000]. In recent years, the research has focused on the integration of different generalization algorithms by modelling the relative dependencies between the operations and the spatial context of the objects (see e.g. [Lamy, Ruas, Demazeau, Jackson, Mackaness & Weibel 1999]). In the context of mobile applications, a combination of relying on pre-computed generalization levels in terms of an MRDB and on-line generalization using web-technology has been proposed (e.g. [Sarjakoski, Sarjakoski, Lehto, Sester, Illert, Nissen, Rystedt & Ruotsalainen 2002])

Progressively transmitting spatial information via a limited bandwidth has been implemented for image formats e.g. in the GIF-Format. Bertolotto & Egenhofer [1999] proposed to also use it for vector data. van Kreveld [2001] presented concepts for the continuous visualization of spatial information in order to reduce the popping effect when discrete changes in the geometry appear. In order to represent different levels of detail of vector geometry, hierarchical schemes can be applied. An example is the GAP-tree for the coding of area partitioning in different levels of detail ([van Oosterom 1995]). The BLG (binary line generalization) tree hierarchically decomposes a line using e.g. the Douglas-Peuker algorithm. In mesh simplification used for the generalization of surfaces, there are methods that animate smooth changes, e.g. by animating the insertion or removal of nodes in a triangular network [Hoppe 1998].

3 Elementary Generalization Operations

The concept of our ideas is explained using the example of the simplification of building ground plans. They can, however, be expanded to other generalization operations, which will be shown later.

3.1 The Generalization Chain

Similar to the ideas introduced by Hoppe [1998] for triangulated meshes, we define for a polygon P consisting of n vertices a maximal representation $P^n \equiv P$, consisting of all original vertices, and a minimal representation P^m, with $m \leq n$ 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.

When a polygon is generalized during pre-processing, one starts from polygon P^n, successively simplifying its representation using generalization operations, 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

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

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 for example the length of the shortest edge in the polygon. Since generalization proceeds using increasing edge lengths, the sequence of ε_j is monotonically increasing. As a first consequence of this, one can pre-compute and record all operations g_j, in order to derive quickly any desired generalization level ε of polygon P by the execution of all generalization operations g_0, ..., g_j, where ε_0, ..., $\varepsilon_j \leq \varepsilon$ 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

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

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.

3.2 Elementary Generalization Operations and Simple Operations

We can define a set of elementary generalization operations (EGO's) such that every generalization chain will be made up of a combination of EGO's. EGO's are intended to be 'meaningful' operations such as 'remove an extrusion'. Each EGO in turn consists of one or more simple operations (SO's) modifying the polygon. SO's are low-level, basic operations, and are the most atomic operations available. It is obvious that there are operations which modify the topology of a polygon, namely the insertion and removal of vertices,

SO	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. Set of simple generalization operations.

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. The parameters involved are edge and vertex id's and relative positions. Edge and vertex id's refer to an implicit numbering of polygon edges and vertices in clockwise or counterclockwise order. 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.

4 Generalization Operations

In this section, the use of the simple operations (SO's) described above is applied to two generalization problems that relate to the generalization of buildings. When presenting buildings in increasingly smaller scale representations, different generalization operations have to be applied, which can be distinguished into three main groups of operations, depending on the scale levels: in scale ranges up to 1:20.000 the individual building shapes are simplified by inspecting the visibility of the shortest facades. This is shown in section 4.1. Further reducing the scale requires an amalgamation of adjacent buildings to form larger objects, as individual buildings would be too small to still be represented. In even smaller scales, buildings have to be selectively removed, enlarged and symbolized. This requires the typification operation, where groups of individual buildings are replaced by new groups with less buildings. This is demonstrated in section 4.2. Thus, in the following, the encoding of two building generalization functions in terms of EGO's will be demonstrated. Amalgamation is not treated here in detail, however in section 6.1 an overview is given, how other generalization operations can be realized.

4.1 Building Generalization

The generalization of building ground plans has to take the regularities of the object into account. Thus, especially rectangularity and parallelism have to be respected and even enforced. We use a rule-based approach which iteratively

inspects small building facades and tries to replace them depending on the spatial context, i.e. its preceding and succeeding building facades [Sester 2000]. In this way, intrusions and extrusions can be eliminated, as well as offsets and corners. Three rules can be decomposed into simple operations. In the following, the generation of the sequence of simple operations is described for each of the three rules, leading to three EGO's. The operations are triggered by a building side s_n that is smaller than a minimum value ϵ, corresponding to a minimum side just visible at a given scale. As noted above, the inverse operation is described, namely going from a generalized version to the more specific one.

Offset

An offset is removed by extending the longer side of the adjacent neighbors of the shortest edge s_n. In Figure 1, s_{n-1} is intersected with s_{n+2} to produce the new node n_{new}. Inverting this process and describing it with the simple operations described above, leads to the following sequence of SO's, composing the EGO for removing offsets.

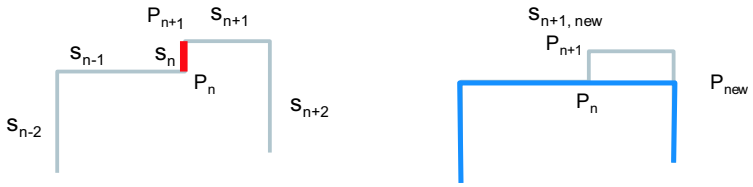

Figure 1. Removing an offset and generating simple operations for it.

1. Note length l_n of the shortest edge s_n:
   `EPS l_n`
2. Insert vertex P_n on side $s_{n+1,new}$ at relative position: RelPos $= \frac{s_{n+1,old}}{s_{n+1,new}}$
   `IV S_n-1, RelPos`
3. Duplicate this vertex, and thus create new point P_{n+1}.
   `DV P_n`
4. Move this new vertex P_{n+1} to the position of the old vertex P_{n+1}, and move vertex P_n to the position of the old vertex P_{n+2} by increments $dx1/dy1$ and $dx2/dy2$:
   `MV P_n+1, dx1, dy1`
   `MV P_n, dx2, dy2`

Extrusion

An extrusion is eliminated by shifting it back or forth to be in line with the corresponding main building side. Figure 2 shows how the extrusion is shifted back to the corpus of the building.

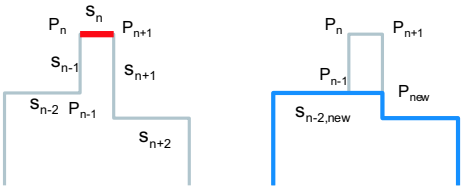

Figure 2. Removing an extrusion and generating simple operations for it.

The extrusion is cut off along the shortest neighboring facade of s_n. In the case of Figure 2, it is s_{n-1}, thus the new structure is derived by intersecting side s_{n-2} with s_{n+1} leading to the intersection vertex P_{new}. Inverting this operation leads to the following sequence of operations, representing an extrusion-EGO:

1. Note length l_n of the shortest edge s_n:
   `EPS l_n`
2. Insert vertex P_{n-1} on side $s_{n-2,new}$ at relative position: RelPos $= \frac{s_{n-2,new}}{s_{n-2,old}}$
   `IV s_n-2, RelPos`
3. Duplicate this vertex, thus creating new vertex P_n
   `DV P_n-1`
4. Move this vertex to the position of the old vertex P_n, and move vertex P_{new} to the position of the old vertex P_{n+1} by increments $dx1/dy1$ and $dx2/dy2$:
   ```
   MV P_n, dx1, dy1
   MV P_new, dx2, dy2
   ```

Corner

A corner is removed by intersecting the adjacent edges (see Figure 3).

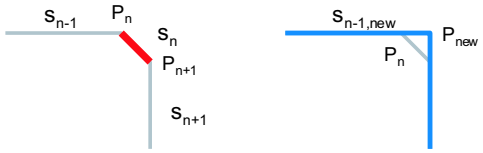

Figure 3. Removing a corner and generating simple operations for it.

A new node is inserted at the intersection point P_{new} and thus creating two new building sides $s_{n-1,new}$ and $s_{n+1,new}$, and removing s_n. Inverting the operation leads to the following sequence:

1. Note length l_n of the shortest edge s_n:
   `EPS l_n`
2. Insert vertex P_n on side $s_{n-1,new}$ at relative position: RelPos $= \frac{s_{n-1,old}}{s_{n-1,new}}$
   `IV s_n-1, RelPos`

3. Move vertex P_{new} to the position of the old vertex P_{n+1} by increments $dx1/dy1$:

```
MV P_new, dx1, dy1
```

Using the EGO's defined above iteratively all facades of the buildings are removed – starting with the shortest one and ending when the removal of one facade leads to the removal of the entire building. Figure 4 shows a sequence of operations in the inverse generalization process. In this case the building is created at an `EPS` of 5, which is the width of building, then an extrusion is created using the SO's specified.

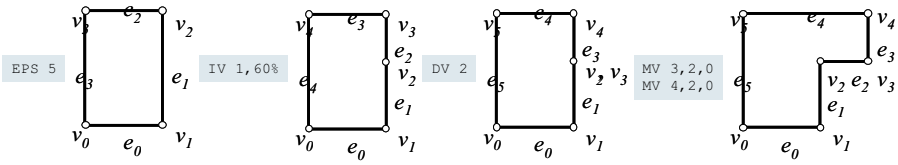

Figure 4. Sequence generating a simple building and corresponding SO's.

In Figure 5, the progressive refinement of four buildings is shown: the snapshots visualize how – at certain stages of the parameter `EPS` that describes the discernable minimal distance – more and more buildings and building details appear. Figure 6 shows some snapshots of a larger area in increasing

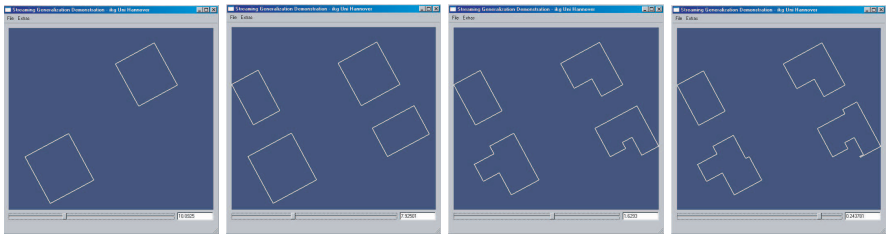

Figure 5. Progressive visualization of four buildings at four levels of detail.

levels of detail: it is clearly visible, how more and more small buildings appear, and they are displayed with more and more detail.

Figure 6. Snapshots of incrementally adding more and more detail.

4.2 Typification

Typification denotes the process of reducing the number of objects out of a set of similar objects while preserving their relative spatial density. Typification is a discrete process, where a set of objects is replaced by another set containing a smaller number of objects. Such a change in representation can be coded with the above described SO's. It has to be realized by removing the objects at one generalization level and introducing new objects at the same time.

The sequence in Figure 7 shows the generation and removal of a building: a new polygon is created at the generalization level `EPS`=10 at position (65,65) by first inserting a vertex; three other vertices are created by duplication of this vertex, then all four vertices are moved to four positions around the center point and thus a square with a side of 20 is finally created. This object 'survives' until the level `EPS`=5 when it 'collapses' by moving all its four corners to the center point again.

```
POLY                    new Polygon
EPS 10                  the following instruction appears at min. distance 10
NPR 65 65               create new polygon ring, consisting of only one point at position 65/65; this
                        point gets ID 0
DV 0                    duplicate this point three times, thus create points 1 to 3
DV 0
DV 0
MV 0 -10 10             move point 0 to the left and upwards
MV 1 -10 -10            also move other points
MV 2 10 -10
MV 3 10 10
EPS 5                   next event happens at minimum value EPS=5
MV 0 10 -10             move points 0 to the right and downwards (i.e. to the original zero-position
                        65/65)
MV 1 10 10              do the same with the other points
MV 2 -10 10
MV 3 -10 -10
```

Figure 7. Example operations to generate and collapse an object.

In this way, a discrete change in representation can be coded. Figure 8 shows the sequence of a typification of a regular structure of 16 buildings being replaced by four and finally by one in the different scales, respectively.

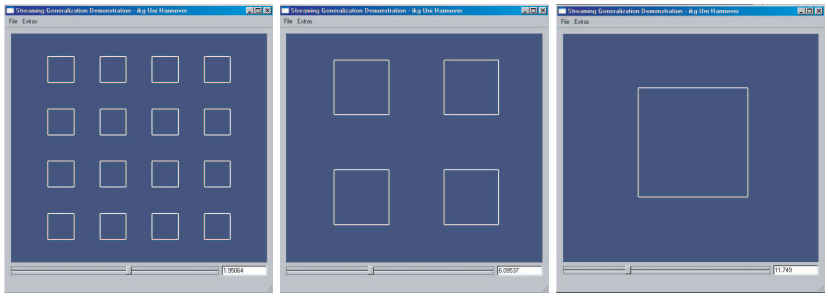

Figure 8. Three different resolutions: high, medium and low, corresponding to $\epsilon = 1.8, 6, 11$, respectively.

The generation of the different representations has to be done in a separate process. In our case, we used an approach based on Kohonen Feature Nets in order to do the rearrangement of the reduced number of objects [Sester 2004].

Figure 9 shows a spatial situation before and after typification. In the process the number of buildings is reduced, the remaining objects are rearranged and then presented by either the original buidlding or a square symbol depending on its size.

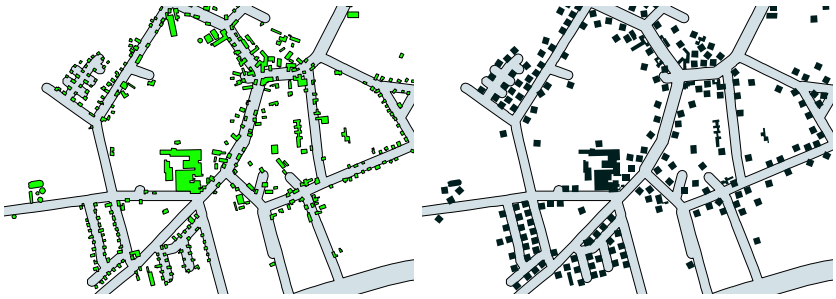

Figure 9. Situation before (left) and after typification of buildings(right).

5 Continuous Generalization and Transmission

5.1 Continuous Generalization

When an object 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 this. Intermediate states can be defined which continuously change the object in response to an EGO. For example, a 'collapse extrusion' EGO (see Figure 2) 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.

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 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. This is implemented in the prototype.

5.2 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 10, 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 generalization level ε, the latter possibly 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.

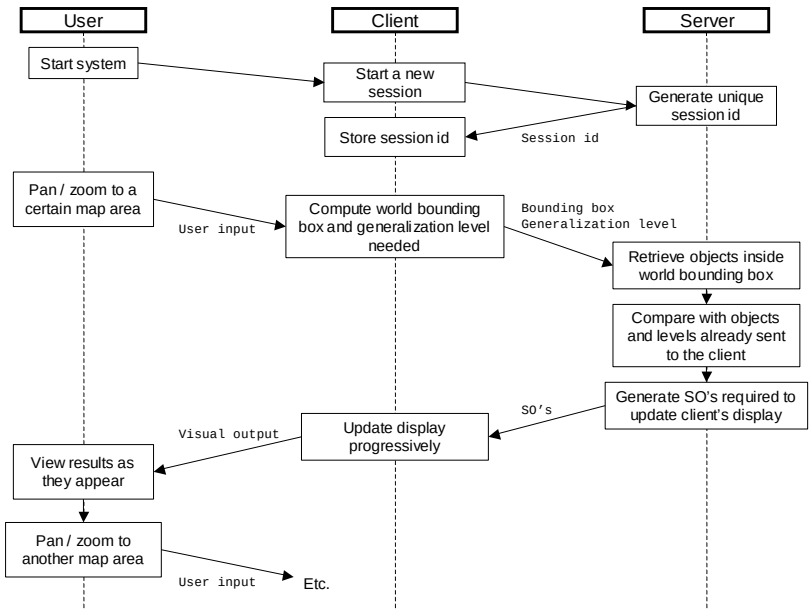

Figure 10. Example interaction diagram for client-server communication.

6 Evaluation of results and discussion

6.1 Discussion on application for other generalization operations

Generalization operations can be characterized by changes occurring to objects which are either *discrete* or *continuous*. These changes can affect *individual objects* and *groups of objects*, respectively.

The two generalization operations described above are of the type of discrete changes affecting individual objects (section 4.1) and groups of objects (section 4.2). What has been shown for these two generalization operations can be extended to other functions as well: amalgamation is an operation that leads to discrete changes of groups of objects, and thus is in the same category as the typification operation. Once an appropriate algorithm is available (e.g. [Bundy, Jones & Furse 1995]), the coding has to be done in terms of SO's and EGO's. In this case, similar to the typification-encoding, objects have to be replaced by other objects (in the case of merging two buildings, the two objects cease to exist and are replaced by the merged one).

Displacement is another important generalization operation to solve spatial proximity conflicts by shifting objects apart and possibly deforming the shapes slightly; it can be categorized as a continuous operation. Displacement can be coded easily as here only a `MV`-operation with respect to all object points is needed. In terms of EGO's a compound movement of all points of an object can be bundled – only however, when no deformation of the object occurs. Existing generalization algorithms can be extended to generate the coding in terms of SO's, e.g. Sester [2004].

6.2 Data volume

In order to evaluate the data volume to be transmitted using the coding scheme proposed above, the following estimation can be done: in our coding scheme, only incremental refinements have to be sent to the client, instead of sending a new representation of the whole data whenever a change in the detail of the data occurs. In case of building simplification a building consisting of n points is iteratively simplified until a polygon with typically 4 points is reached and then it vanishes. In each of the EGO's one or two points are removed, thus in order to present all possible LOD's approx. $n/2$ different representations have to be transmitted, leading to $n * n/2$ points. Our coding scheme, however, requires only $n/2$ operations leading to a reduction factor of approx. $1/n$.

The representation in our coding scheme is compared to the original representation of the data set in terms of an ESRI shapefile for different data sets: the size of the data set in Figure 6 consisting of 119 buildings is 30Kbyte; the original representation in terms of the ESRI shapefile (the .shp-file) is 20 kByte. Another dataset consisting of 2000 buildings requires 0.45 MByte in terms of SO's. The original shapefile needs 0.312 MByte. The following observations have to be considered when evaluating these numbers:

- There have not yet been any considerations concerning efficient storage of the code.
- The SO-coding includes *all* generalization levels that are possible between the most detailed representation of the objects and its most general one.
- The coding in terms of EGO's allows to group several SO's to a higher level standard operation. In the current implementation, however, the EGO's not yet have been efficiently coded, but are described in terms of SO's.

7 Conclusion and Future Work

The research presented in this paper was motivated by the fact that a spatial data set on a small display device like a PDA typically has to be visualized in different levels of detail: for an overview only coarse information of a larger extent of the map is needed; when the user zooms in in order to inspect details, progressively new information is loaded for that smaller section of the map the user is interested in. However, the approach is generally applicable, when spatial data to be presented is not available on the device itself, but has to be transmitted from a remote server. Besides the transmission to mobile devices described above, this also holds for the presentation of spatial data in the internet.

In the paper an approach was presented that decomposes generalization operations into simple operations leading to changes in topology and geometry. It was shown how these simple operations can be used to code more complex operations dealing with continuous and also discrete changes in the objects. Finally, an implementation for a client-server data transmission was presented. In the current concept, the generation of the EGO's is an off-line process, that is done beforehand. It is, however, also possible to generate the code on-the-fly upon a request of the client. In this case, the response time on the client will be delayed by the time needed for the generalization process. Thus the trade-off between storing overhead of the data in terms of EGO's and processing time for the generalization has to be balanced. This will be subject to further investigations. Future work will also concentrate of implementing this concept in a distributed system environment and on the integration of other generalization operations.

References

Bertolotto, M. & Egenhofer, M. [1999], Progressive Vector Transmission, *in:* 'Transactions of the ACMGIS99', Kansas City, MO, pp. 152–157.

Bundy, G., Jones, C. & Furse, E. [1995], *Holistic generalization of large-scale cartographic data*, in Müller, Lagrange & Weibel [1995], pp. 106–119.

Douglas, D. & Peucker, T. [1973], 'Algorithms for the reduction of the number of points required to represent a digitized line or its caricature', *The Canadian Cartographer* **10**(2), 112–122.

Hake, G., Grünreich, D. & Meng, L. [2002], *Kartographie*, Gruyter.

Harrie, L. E. [1999], 'The constraint method for solving spatial conflicts in cartographic generalization', *Cartographiy and Geographic Information Science* **26**(1), 55–69.

Højholt, P. [1998], Solving Local and Global Space Conflicts in Map Generalization Using a Finite Element Method Adapted from Structural Mechanics, *in:* T. Poiker & N. Chrisman, eds, 'Proceedings of the 8th International Symposium on Spatial Data handling', Vancouver, Canada, pp. 679–689.

Hoppe, H. [1998], Smooth view-dependent level-of-detail control and its application to terrain rendering, *in:* 'IEEE Visualization '98', pp. 35–42.

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, pp. 1225–1234.

Müller, J.-C., Lagrange, J.-P. & Weibel, R., eds [1995], *GIS and Generalization - Methodology and Practice*, Taylor & Francis.

Nivala, A.-M., Sarjakoski, L., Jakobsson, A. & Kaasinen, E. [2003], Usability Evaluation of Topographic Maps in Mobile Devices, *in:* 'Proceedings of the 21st International Cartographic Conference of the ICA', Durban, South Africa, pp. 1903–1913, CD–ROM.

Regnauld, N. [1996], Recognition of Building Clusters for Generalization, *in:* M. Kraak & M. Molenaar, eds, 'Advances in GIS Research, Proc. of 7th Int. Symposium on Spatial Data Handling (SDH)', Vol. 1, Faculty of Geod. Engineering, Delft, The Netherlands, pp. 4B.1–4B.14.

Reichenbacher, T. [2004], Mobile Cartography: Adaptive Visualization of Geographic Information on Mobile Devices, PhD thesis, Technische Universität München.

Sarjakoski, T., Sarjakoski, L., Lehto, L., Sester, M., Illert, A., Nissen, F., Rystedt, R. & Ruotsalainen, R. [2002], Geospatial Info-mobility Services - a Challenge for National Mapping Agencies, *in:* 'Proceedings of the Joint International Symposium on 'GeoSpatial Theory, Processing and Applications' (ISPRS/Commission IV/SDH2002)', Ottawa, Canada, CD-Rom.

Sester, M. [2000], Generalization based on Least Squares Adjustment, *in:* 'IAPRS', Vol. 33, ISPRS, Amsterdam, Holland.

Sester, M. [2004], Optimizing Approaches for Generalization and Data Abstraction, Technical report, accepted for publication in: International Journal of Geographical Information Science.

van Kreveld, M. [2001], Smooth Generalization for Continuous Zooming, *in:* 'Proceedings of the ICA', Beijing, China.

van Oosterom, P. [1995], The GAP-tree, an approach to 'on-the-fly' map generalization of an area partitioning, *in* Müller et al. [1995], pp. 120–132.