Continuous Generalization for fast and smooth Visualization on small Displays

Monika Sester & Claus Brenner

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

Commission IV, WG IV/3

KEY WORDS: Cartography, Generalization, Mobile, Real-time, Visualisation

ABSTRACT:

With the increasing availability of small mobile computers, there is also an increasing demand for visualizing cartographic objects on those devices. Prominent applications are location based services in general, and car and pedestrian navigation in particular. In order to be able to offer both detail and overview of a spatial situation, the devices have to provide flexible zooming in and out in real-time. The presentation of spatial data sets in different zoom levels or resolutions is usually achieved using generalization operations. When larger scale steps have to be overcome, the shape of individual objects typically changes dramatically, also objects may disappear or merge with others to form new objects. As theses steps typically are discrete in nature, this leads to "popping effects" when going from one level of detail to the other.
In this paper, we will describe an approach to decompose generalization methods into elementary operations that can then be implemented in a continuous way. For example in the case of displacement, an object will not simply jump from one position to the other, but slowly shifted from its original position to the new one. In the case of simplification of building ground plans, the elementary operations e.g. care for removing extrusions or intrusions of buildings, as well as offsets. In the paper we will identify elementary generalization operations and also present their implementation as a continuous operations. We will apply these concepts for line generalization, the generalization of building ground plans and for displacement.

1. INTRODUCTION AND OVERVIEW

The presentation of spatial data in different levels of detail is a basic requirement in order to be able to fully understand spatial processes. In cartography it has traditionally been accounted for by the series of topographic maps (e.g. different scales from 1:10.000 to 1:1 Million). For their production, generalization operations are being applied that generate a new representation from the given detail data.

The need for presenting spatial data in different resolutions recently came up again from a completely new domain: in order to present spatial information on small mobile displays – typically user location or navigation instructions – there is a strong need for generalization, because on the small displays only a reduced information content can be displayed at a time. As the small display devices typically do not dispose of large storage capabilities for storing digital data sets at different resolutions, the need for efficiently transmitting the spatial information arises.

This was the basis for this research, that aims at developing a method for incrementally transmitting more and more information in terms of object details to a small mobile device through a possibly limited bandwidth channel by incremental streaming. When a user inspects spatial data using a mobile client, first only the coarsest information is transferred to give an overall impression. Then, objects in the vicinity of the user will be incrementally loaded, until – if the user wishes so – the whole scene is given at the highest level of detail available.

The idea is to pre-compute a sequence of vector representations at different levels of detail, which are then incrementally sent to the client. These different representations, in our case, are coded efficiently in terms of a set of simple operations. These operations can be generated by appropriate adaptation of existing generalization operations. The procedure provides methods to visualize and animate these changes in a continuous and streaming fashion.

The paper is organized as follows: After an analysis of demands for progressive information transmission, a brief classification of generalization algorithms is given. Then, the elementary operations to code incremental changes of objects are presented. Different generalization functions are adapted in order to produce a representation in terms of those simple operations. A summary concludes the paper.

2. RELATED WORK AND DEMANDS FOR PROGRESSIVE INFORMATION TRANSMISSION

The basic demand for continuous generalization is that the changes occurring when going from one representation to the next are small enough in order not to be noticed. Thus, the user is not disturbed by coarse visible changes like object parts popping up or objects suddenly disappearing.

In order to provide such a smooth transition from one scale to the next, incrementally representations with more and more detail have to be visualized. This would imply that a very dense series of different representations is generated and has to be transmitted to the user while he/she is zooming in or out.

Besides high demands for the storage of those large number of representations on the server, this also has high requirements concerning the transmission of the data, as a large number of potentially large data sets has to be transmitted. Due to the fact that changes in the data occur only at selected places in the data sets, potentially also highly redundant data is sent. Alternatives are to provide only a limited set of representations, where large changes occur – comparable to the map series of topographic maps. The project GiMoDig aims at providing a combination of on-line generalization and access to pre-generalized data [Sarjakoski et al., 2002]. Bertolotto & Egenhofer [2001] describe an approach for progressively transmitting vector data by pre-computing a sequence of map representations at different Levels of Detail (LoDs). Between adjacent scales, appropriate interpolations or morphing operations can be done ([Cecconi et al., 2002], [van Kreveld 2002]).

Another alternative is to sent only changes or differences in the data set, which already can reduce the amount of data considerably. This is e.g. well known from the progressive transmission of GIF-images over the internet. Thiemann [2002] proposes to use this method for the visualization of 3D building data in different levels of detail. A further possibility is not to send the changes as such, but a set of operations that describe the object and the changes. This requires that on the client side these instructions can be interpreted in order to correctly restore the object. A basic requirement for the coding scheme is that the amount of data to be transmitted should be less than that would be needed to transmit the original data set.

In order to represent different levels of detail of vector geometry, hierarchical schemes can be used. On example is the GAP-tree for the coding of area partitionings 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 [Douglas & Peucker, 1973].

In our approach a set of elementary operations was defined that allow for describing geometric and topologic changes in vector data sets. Generalization operations can be decomposed into a sequence of operations leading to a sequential reduction / increase of detail. Thus data coded in a vocabulary of so called Simple Operations (SO's) can be sent to the client, where it is restored again in order to be visualized.

3. 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. In the following examples for both types of changes are given.

3.1 Discrete changes of individual objects

Changes of this kind can be characterized by the fact that the topology and the geometry of the object changes. Examples for this class of changes are operations like the simplification of building ground plans or simplification of lines (point reduction) using e.g. Douglas-Peuker filtering.

3.2 Continuous changes of individual objects

Continuous changes of objects occur when the topology remains the same, however geometry changes by shifting either the whole object or individual points of the object. Displacement is a typical representative for such a change. Also in the case of enlargement, only the position of object vertices changes, not affecting the topological structure of the objects.

3.3 Discrete changes of groups of objects

These changes typically occur when larger scale changes have to be traversed and thus the aggregation level and often the type of object changes. An example is typification, where a group of objects is represented by a new group consisting of less objects.

The following table gives a classification of the different generalization operations into the different categories. Based on this analysis examples for the implementation of the operations are given. The operations highlighted in bold are described in more detail.

Operation	*Discrete, individual (3.1)*	*Discrete, group (3.3)*	*Continuous (3.2)*
Simplification (e.g. Gauss)			X
Point reduction (e.g. Douglas-Peuker)	**X**		
Building generalization	**X**		
Displacement			**X**
Typification		**X**	
Aggregation		X	X
Enhancement (e.g. enlargement)			X

4. DECOMPOSITION OF CHANGES INTO ELEMENTARY OPERATIONS

The coding scheme has been developed for the generalization of polygons, especially building ground plans. It is, however, obvious, that the approach is generally applicable to all the above mentioned generalization operations.

4.1 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 $m \le n$ 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 in case of a building 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 in Section 5.2, 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 later – 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^{ij}. 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 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 the minimal representation P^m followed by a sufficient number of inverse generalization operations.

4.2 Encoding Elementary Generalization Operations

We call the generalization operations we introduced above 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 1 shows how SO's combine to an inverse EGO, which realizes the creation of an extrusion of a building annex. Starting from the top 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 lower right. It can be observed that the numbering of the nodes and lines is continuously adjusted in order to preserve the correct sequence. 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.

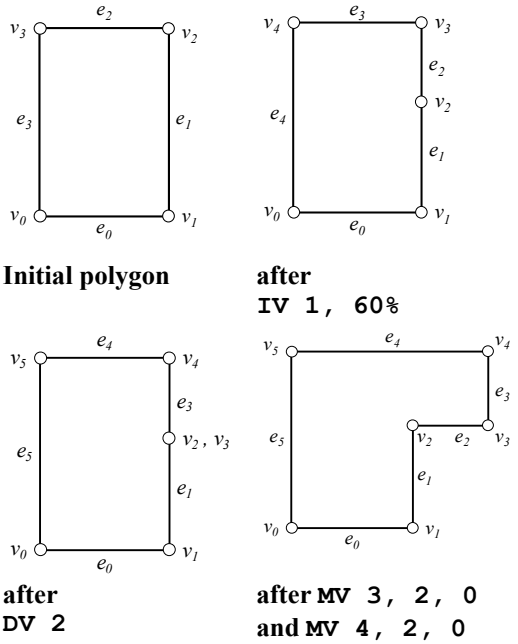


Figure 1: Example for an inverse EGO, forming an L-shaped building from a rectangular building. The EGO is decomposed into four SO's.

5. REALIZATION AND EXAMPLES

In the following, we will demonstrate how several generalization operations can be adapted to generate a sequence of simple operations described above in order to generate a generalization chain that can incrementally be sent to a client.

5.1 Line simplification / Point reduction

Simplifying lines or polygon outlines can be accomplished using filtering techniques or point reduction methods. For point reduction, different algorithms have been developed that either locally, regionally or globally investigate a line and decide upon which point can be omitted. The most popular algorithm is the globally acting Douglas-Peuker algorithm. In order decompose the point reduction process into a sequence of reversible elementary generalization operations, the following considerations can be made. Similar to generating a BLG tree, a scale dependent decomposition of a line, is generated by recursively extending the levels of detail in a tree structure. At the root of the tree is the most coarse line consisting of start and endpoint (see Figure 2). The inner nodes represent intermediate generalization levels specifying line sectors with an associated generalization level, and the leaves, finally, contain the original line elements (here, the associated generalization level is obviously 0). The generalization level or scale in this case is directly related to the distance of that point from the corresponding base line. For example in sector AF, at scale level *c* a split of the line into the two sectors AC and CF will be

done. In order to present a certain level of detail, the tree has to be traversed down to the given scale level.

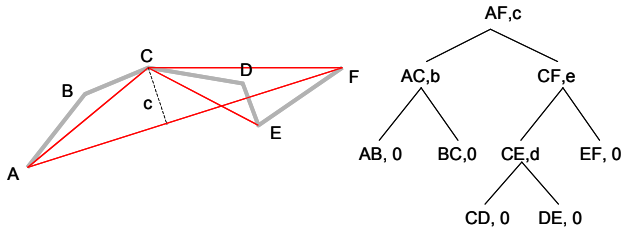


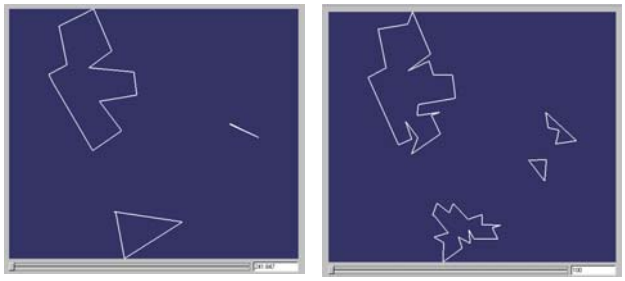
Figure 2: Original line (left) and corresponding BLG-tree (right); the scale levels are indicated in the nodes of the tree. At the leaves, the scale levels are zero.

For the generation of the BLG-structure, the whole tree has to be generated in order to give the full zooming from coarse to fine. The transformation into the SO's is straightforward (confer Table 1): Starting point is a new line which appears at a certain scale level EPS, that corresponds to the length of the line s_{AF}. The line is generated by creating point 0 at position A (NPR), duplicating this point (DV) and moving it to the position F by increments dx_{AF}, dy_{AF}. (MV). At scale level c a new point is inserted (EPS c). This is accomplished by duplicating point A (i.e. point 0 in the internal number scheme) and moving it to position c by increments dx_{AC}, dy_{AC}. All this information is directly coded in the tree. The only issue is an appropriate sequencing of the insertion of the points, taking the respective scale levels of the nodes into account.

POLY	Create new object
EPS s(AF)	scale level EPS = distance between points A and F
NPR xA yA	Create point 0 with coordinates xA and yA
DV 0	Duplicate this point -> create point 1
MV 1 (xF-xA) (yF-yA)	Move point 1 by dx and dy -> move it to point F
EPS c	New event at distance c
DV 0	Create new point after point 0 by duplicating point 0
MV 1 (xC-xA) (yC-yA)	Move this point by dx/dy to point C
…	…

Table 2: Coding Douglas-Peuker line simplification.

Figure 3 presents some screenshots of the successive refinement of polygons using the SO-coding. The iterative refinement is clearly visible; the user can control the level of detail with the slider below. Furthermore, the transmission is organized in a way that only data in the current view will be loaded and refined.



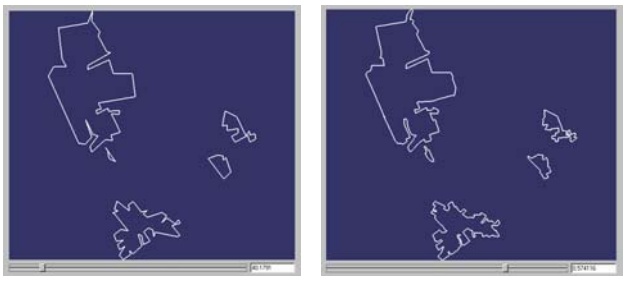


Figure 3: Screenshots visualizing increasing refinement of the polygon-visualization (from top left to lower right).

The following Figure 4 shows that the Douglas-Peuker algorithm is not appropriate for the generalization of structured objects such as buildings. Therefore, in the next section, an algorithm for building generalization and the corresponding decomposition into SO's is presented.

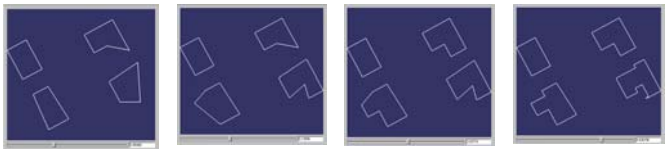


Figure 4: Sequence of images of using DP-algorithm to building generalization – which is obviously not suited for the generalization of such structured objects.

5.2 Building simplification

Building simplification is a special case of a point reduction method, where the specific properties of these objects are taken into account. In this case the point reduction is more a structure reduction, as properties like parallelism and rectangularity have to be respected in the algorithm. Here, we used a method that analyzes the shape of the building and defines appropriate methods to eliminate too small parts of the ground plan, i.e. too short façade elements (see [Sester 2000]). Three different kinds of structures can be identified, for which appropriate reduction methods are defined: extrusion or intrusion, offset, and corner (see Figure 5).

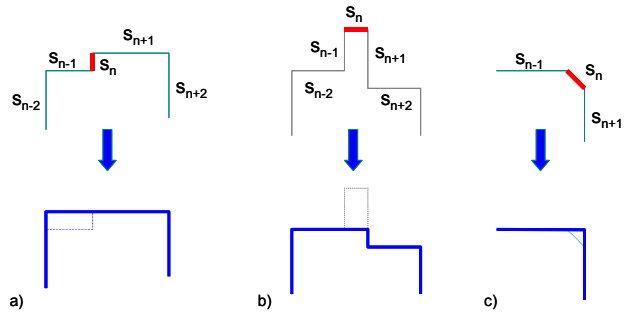


Figure 5: Elimination of short facade sn: offset, intrusion/extrusion and corner.

The decomposition of the changes in terms of SO's is straightforward. For the example of the offset it is the following:

An offset consisting of 4 points is replaced by a straight line consisting of 2 points (see Figure 5a). The reduction process – which is done when eliminating or generalization this structure – extends the longest edge adjacent to the short edge s_n, in this case it is line s_{n+1}. A new point is created at the intersection of the extended line and the predecessors predecessor line (in this case between line s_{n+1} and line s_{n-2}). In order to code this process in terms of SO's, it has to be inverted, i.e. we start from the end situation with one line between points 1 and 4, then

insert point 2 on this line at 33% of the line length, then duplicate this point in order to get point 3. Moving points 1 and 3 new to their final position ends the process. In summary, the EGO for an offset is the following:

```
…
IV 1 2 0.33
DV 2
MV 1 dx dy
MV 3 dx dy
….
```

In a similar way also the generalization operations for the other two events can be coded (for more details see [Brenner & Sester, 2003]). Figure 6 shows an example for the successive presentation of more and more details for four buildings (compare to Figure 4, where not appropriate Douglas-Peuker algorithm was used). Figure 7 shows some screenshots of a larger area of a city.

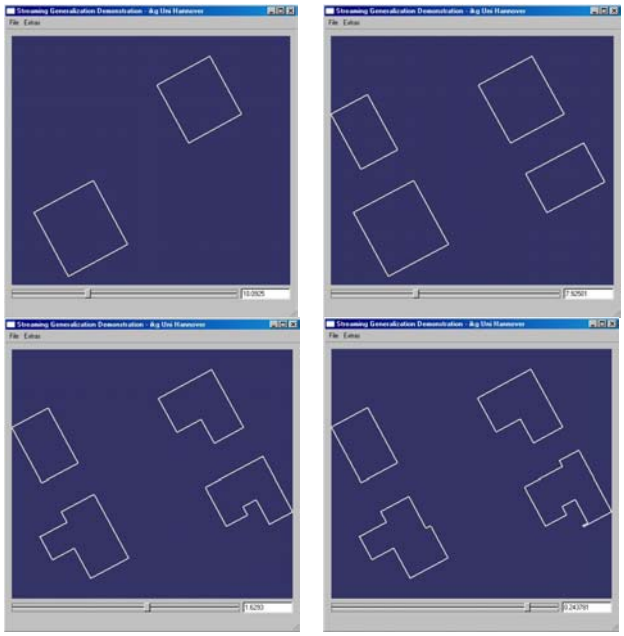


Figure 6: Presentation of four buildings in different levels of detail.

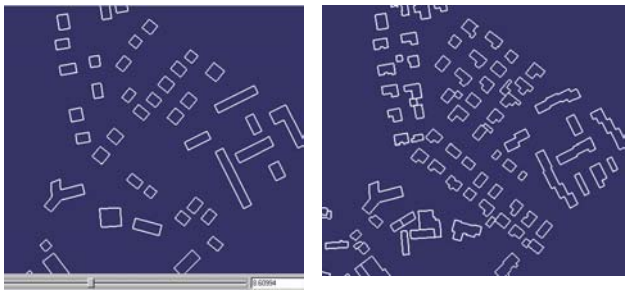


Figure 7: Two screenshots with different generalization levels of buildings in city.

5.3 Typification

Typification involves that a group of objects is replaced by a new group with less objects. This means, that extreme changes occur between the different representations, as objects are eliminated and replaced by new ones. Coding this process in terms of EGO's is simple: an object collapses and a new object emerges. This involves that a new geometry is created.

5.4 Displacement

The coding of the displacement operation in terms of SO's is very simple, as it only consists of move-operations (`MV`) of the original points to their new positions. We use a least squares adjustment based approach for calculating the displacement between all objects in a scene ([Sester 2004]). Figure 8 shows an example for a spatial situation before and after displacement: it is obvious, that only in case where conflicts occur (red areas), the objects change both position and also (partly) their shape (this is indicated in different shades of green in Figure 8). The resulting translations in the individual objects are coded in terms of SO's.

Figure 8: Displacement: Overlay of original objects in conflict (top) and solution after automatic displacement.

5.5 Coding efficiency

In order to compare the storage requirements of the coding in terms of EGO's with the full presentation of several generalized instances of the object, the following estimation can be made. It is done in detail for the case of **point reduction**, but can be extended to the other operations mentioned here as well.

A line consisting of *n* points is reduced to 1 point and then vanishes, or vice versa it comes into existence with 1 point and then iteratively is refined by including new points until its detailed structure is achieved. This means, that in the original representation *n* double values (x and y) have to be stored. Transmitting all the possible n representations would require

$$1 + 2 + 3 + \ldots + n\text{-}1 + n = \tfrac{1}{2}\, n\,(n+1) \text{ Points}$$

or twice the number of double values in terms of coordinates. Thus, the amount of data to be transmitted is in the order of n^2.

Storing this information in terms of SO's requires two operations for each intermediate point (`DV` <int>, `MV` <float> <float>), which requires

n points or 2*n coordinate differences

In this case, float values can be used, as the coordinate differences <dx,dy>-values are typically small. In addition to

the points, also the operation codes (`IV, DV`, …) together with integer values indicating point id's have to be coded. Altogether, this is in the order of *n* which basically means that all representations of an object can be transmitted for the price of transmitting one.

Coding **displacement** is more demanding concerning the data volume, as it requires the same number of coordinates, as the points are only moved. However, the numbers are small, as the movements of the points are typically very small compared to the large coordinate values, that need double precision values. Furthermore, as only changes are encoded, not the whole data set has to be transferred in all scale-steps. Finally, also an operation could be defined, that encodes the movement of an object as a whole. During **typification** the objects are replaced by new objects, i.e. completely new objects are created. Thus, no incremental change from the old situation to the new one can be done, which has the consequence that the full object representation has to be created and hardly a reduction in volume can be achieved.

6. 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 5b) 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. It is realized by decomposing the movement into a number of intermediate steps that give the impression of smooth changes. For more details see [Brenner & Sester, 2003].

7. SUMMARY

An approach was presented to decompose changes in object geometry into a small set of simple operations. These operations can express the creation of objects as well as iterative refinement of their shapes. This coding scheme was used to represent different generalization levels of objects efficiently. For different generalization operations it could be shown, how this representation was generated. A comparison concerning the storage and coding demands with respect to representing the full geometry was made and it was shown that a reduction in the amount of data to be transmitted by approximately the factor n can be achieved. Besides incrementally presenting the iterative changes in the geometry, it was also shown that the changes can be animated, leading to nearly invisible changes between the different representations when changing the scale.

8. REFERENCES

Bertolotto, M., and Egenhofer, M., [2001], Progressive Transmission of Vector Map Data over the World Wide Web, GeoInformatica - An International Journal on Advances of Computer Science for Geographic Information Systems, Vol. 5 (4), Kluwer Academic Publishers, pp.345-373

Brenner, C. and Sester, M. [2003]: Continuous Generalization for Small Mobile Displays, International Workshop on Next Generation Geospatial Information, October 19-21, 2003, Cambridge (Boston), Massachusetts, USA.

Cecconi, A., Weibel, R. & Barrault, M., [2002]. Improving automated generalization for on-demand web mapping by multiscale databases. *10th International Symposium on Spatial Data Handling*, Ottawa, Canada.

Douglas, D. H. & Peucker, T. K., [1973]. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*,10, pp. 112-22.

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.

Oosterom, P. van [1995]: "On-the-Fly" Map Generalization of an Area Partitioning. In: GIS and Generalization, Methodology and Practice. Editors J.C. Müller, J.P. Lagrange and R. Weibel. Taylor & Francis, London, pages 120-132, 1995.

Sarjakoski, T, L.T Sarjakoski, L. Lehto, M. Sester, A. Illert, F. Nissen, R. Rystedt, and R. Ruotsalainen [2002]: 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, 5p, CD-ROM.

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 [2004]: Optimizing Approaches for Generalization and Data Abstraction, accepted for publication in: International Journal of Geographic Information Science.

Thiemann, F. [2002], Generalization of 3D building data, IAPRS Vol. 34, Part 4, "GeoSpatial Theory, Processing and Applications", Ottawa, Canada.

ACKNOWLEDGEMENT

This research is part of the GiMoDig project, funded by the European Union, IST 2000, 30090, and the VolkswagenStiftung.