

# A vocabulary for a multiscale process description for fast transmission and continuous visualization of spatial data

Monika Sester & Claus Brenner

*Institute of Cartography and Geoinformatics, Leibniz Universität Hannover,  
Appelstraße 9a, 30167 Hannover, Germany*

---

## Abstract

With the increasing availability of small mobile computers there is also an increasing demand for visualizing spatial data 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 same demands arise from the increasing amounts of data available and accessible by web services through limited bandwidth channels. 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 these steps typically are discrete in nature, this leads to visual ‘popping effects’ when going from one level of detail to the other.

In this paper, we will present an approach to decompose generalization into simple geometric and topologic operations that allow describing the complete generalization chain to generate a multiscale object representation. The goal is to generate a representation without redundancy and to transmit only that information which is needed when scale changes occur. This representation scheme ultimately also enables a continuous visualization, where the changes between the representations are visually indistinguishable. We identify elementary generalization operations and apply these concepts for polyline simplification, the generalization of building ground plans and for displacement.

*Key words:* Cartography, Generalization, Streaming, Visualisation

---

---

*Email address:* [monika.sester,claus.brenner@ikg.uni-hannover.de](mailto:monika.sester,claus.brenner@ikg.uni-hannover.de)  
(Monika Sester & Claus Brenner).

*URL:* [www.ikg.uni-hannover.de](http://www.ikg.uni-hannover.de) (Monika Sester & Claus Brenner).

## 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 coarse representations from a given detailed data set.

The need for presenting spatial data in different resolutions recently arose 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 visualized at a time. As the small display devices typically do not dispose of large capabilities for storing digital data sets at different resolutions, the need for efficiently transmitting the spatial information from a remote server is evident. The same is true for large data sets accessed via the internet.

This problem was the starting point of our research, which 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 or internet client, first only the coarsest information is transferred to give an overall impression. Then, objects in the zooming area 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. These different representations, in our case, are coded efficiently in terms of a set of simple operations, describing topologic and geometric changes. These operations can be generated by an appropriate adaptation of existing generalization operations. This code is incrementally sent to the client, where it has to be restored and visualized.

The paper is organized as follows: After a review of related work and 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 and an outlook on future work conclude the paper.

## 2 Related work and demands for progressive information transmission

The basic requirement for progressive data transmission is that the changes occurring when going from one representation to the next are small enough in order not to be visually 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, incremental object representations with more and more detail have to be visualized. This would imply that a very dense series of different scale representations is generated that has to be transmitted to the user while he/she is zooming in or out. Besides high demands for the storage of that 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 scale changes of individual objects are not homogeneously distributed in the whole data set, potentially also highly redundant data is sent, i.e. an object is sent again, even if it has not changed from the previous scale. An alternative is to provide only a limited set of representations at dedicated scale levels comparable to the map series of topographic maps. The project GiMoDig aimed at providing a combination of on-line generalization and access to pre-generalized data (Sarjakoski et al. (2002)). Bertolotto and Egenhofer (1999) describe an approach for progressively transmitting vector data by pre-computing a sequence of map representations at different Levels of Detail (LoDs). Another possibility is to send only changes or differences in the data set, which already can reduce the amount of data considerably. Such a mechanism is 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 similar approach is given by Yang et al. (2007) for the point reduction operation: they present a scheme for an incremental vertex decimation taking also topological consistency of neighboring objects into account. Between adjacent scales, interpolations or morphing operations can be applied in order to provide a visually smooth transition (van Kreveld (2001), Cecconi et al. (2002), Nöllenburg et al. (2008)).

For the representation of different levels of detail of vector geometry hierarchical schemes can be used. One example is the GAP-tree for the coding of area partitions in different levels of detail (van Oosterom (1995)). This data structure also allows for a progressive data transfer (van Oosterom (2005)). The BLG (binary line generalization) tree hierarchically decomposes a line using e.g. the Douglas-Peucker algorithm (Douglas and Peucker (1973)). Ai et al. (2004) describe a hierarchical decomposition of objects using a series of convex hulls.

A further option 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.

In our approach a set of elementary operations is defined that allow for describing geometric and topologic changes in vector data sets. Generalization operations can be decomposed into a sequence of elementary operations leading to a sequential reduction and increase of detail when zooming out or in, respectively. Thus, data coded in a vocabulary of so called Simple Operations (SOs) can be sent to the client, where it is restored again in order to be visualized. Due to the multi-scale property of this coding scheme, only that amount of detail has to be sent which is required by the user. The user can stop the transmission as soon as enough information for the current purpose has been obtained. The system consists of three parts: an off-line pre-processing step that generates the multi-scale code using generalization functions, the transmission of the code to the client, and a process that is able to recover all the intermediate generalization levels in the client. The basic principle of the coding scheme was described in an earlier paper (Brenner and Sester (2005)). In this presentation it is applied to new generalization functions and the coding efficiency is discussed.

### 3 Generalization operations

In recent years, advancements in the automation of generalization operations can be observed. For a comprehensive overview on automatic generalization see Mackaness et al. (2007). 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. They result in changes in topology and/or in geometry. In the following, examples for these types of changes are given.

#### 3.1 *Discrete changes of individual objects*

This situation is characterized by the fact that the topology and the geometry of the object changes. Examples for this class of changes are point reduction operations like the simplification of building ground plans or simplification of lines using e.g. Douglas-Peucker filtering. Another example is symbolization, where an object is replaced by a new geometry or a symbol. Finally, also the collapse operation can be classified into this category, as the original geometry is replaced by a completely new geometry, e.g. a polygon is replaced by a line or a point.

### 3.2 *Discrete changes of groups of objects*

This type of change typically occurs when larger scale ranges have to be traversed and thus the abstraction 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 fewer objects (Müller and Wang (1992); Regnauld (1996); Sester (2007)). Another example is the amalgamation where nearby objects are merged to a new object.

### 3.3 *Continuous changes of individual objects*

Continuous changes of objects occur when the topology remains the same, however geometry changes by moving either the whole object or individual points of the object. Displacement is a typical representative for such a change. Algorithms based on continuous optimization have been developed (Højholt (1998); Harrie (2001); Sester (2005)). Also in the case of the enlargement operation only the positions of object vertices change, not affecting the topological structure of the objects. The same is true for continuous simplification of objects, e.g. Gaussian smoothing of lines, where the original object points are relocated.

### 3.4 *Classification of operations*

Table 1 gives a classification of generalization operations into these different categories according to the typology presented by Regnauld and McMaster (2007). Based on this analysis, examples for the implementation of operations are described in more detail in Section 5, namely for the operations highlighted in bold.

## **4 Decomposition of generalization operations into elementary operations**

A coding scheme has been developed for the generalization of polygons, especially building ground plans. As it is able to describe topologic and geometric changes it is generally applicable to all the above described generalization operations.

<i>Operation</i>	<i>Discrete individual (Section 3.1)</i>	<i>Discrete, group (Section 3.2)</i>	<i>Continuous (Section 3.3)</i>
Smoothing (e.g. Gaussian smoothing)			X
<b>Simplification: Point reduction</b>	<b>X</b>		
<b>Simplification: Building generalization</b>	<b>X</b>		
Aggregation		X	X
Amalgamation		X	X
Collapse	X		X
Refinement / Symbolization	X		
Exaggeration (e.g. enlargement)			X
<b>Displacement</b>			<b>X</b>
<b>Typification</b>		<b>X</b>	

Table 1

Typology of generalization operations (after Regnauld and McMaster (2007)).

#### 4.1 The Generalization Chain

Similar to the ideas introduced by Hoppe (1996) for triangulated meshes, 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. In the case of a building this can be a rectangle, with  $m = 4$ , or the empty polygon, with  $m = 0$ . Map generalization starts from polygon  $P^n$ , successively simplifying its representation using generalization operations (as described in Section 5) and 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  $i_0 = n, i_1, \dots, i_k = m$ , then a sequence of generalized polygons

$$P \equiv P^n \equiv P^{i_0} \xrightarrow{g_0} P^{i_1} \xrightarrow{g_1} \dots \xrightarrow{g_{k-1}} P^{i_k} \equiv P^m \quad (1)$$

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  $\epsilon_j$ , which relates to the display scale and can be – as discussed later – for example the length of the shortest edge in the polygon. Since generalization proceeds using increasing edge lengths, the sequence of  $\epsilon_j$  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  $\epsilon$  by the execution of all generalization operations  $g_0, \dots, g_j$ , where  $\epsilon_0, \dots, \epsilon_j \leq \epsilon$  and  $\epsilon_{j+1} > \epsilon$ .

However, for an incremental refinement of data, the inverse operations  $g_j^{-1}$  are more interesting, producing a more detailed polygon from a generalized one. Thus, we have the sequence

$$P \equiv P^m \equiv P^{i_k} \xrightarrow{g_{k-1}^{-1}} P^{i_{k-1}} \xrightarrow{g_{k-2}^{-1}} \dots \xrightarrow{g_0^{-1}} P^{i_0} \equiv P^n \quad (2)$$

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

#### 4.2 Encoding elementary generalization operations in a basic vocabulary

We call the generalization operations  $g_j(\epsilon_j)$  which we introduced above Elementary Generalization Operations (EGOs), because every generalization chain will be made up of a combination of EGOs. Each EGO in turn con-

sists of one or more Simple Operations (SOs) modifying the polygon. Thus, we can think of the SOs to be the basic building blocks, whereas EGOs can be seen as being application specific macro instructions. For example, generating an extrusion of a building by adding and moving a set of points would be an EGO, while moving an individual point would be a SO. It is obvious from the discussion in Section 3 that both operations which modify the topology of a polygon and operations which affect only the geometry are required: topology related operations insert and remove vertices, whereas geometric operations modify the geometry by altering coordinates. Table 2 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 may 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.

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 2

Set of simple generalization operations (SOs).

Besides the simple operations given in Table 2 there are additional operations for specifying the scale level at which a change occurs (**EPS value**) and the creation of a new polygon, starting with a single vertex at position *x/y*: **NPR x y**.

A client-server architecture and a communication scheme to stream the geometries in terms of SOs and to interpret and reconstruct them on the client has been developed, which is described in detail in Brenner and Sester (2005).<sup>1</sup> In this implementation the aggregation in terms of EGOs has not yet been realized, thus the code is based on SOs only. In the following section 5 we do, however, describe how EGOs could be described as compositions of individual SOs.

<sup>1</sup> A demonstration program of the streaming generalization approach can be downloaded from the website of the authors ([http://www.ikg.uni-hannover.de/forschung/vw\\_stiftung/projekte/software.html](http://www.ikg.uni-hannover.de/forschung/vw_stiftung/projekte/software.html))



## 5 Realization of generalization operations and examples

In the following, we will demonstrate how generalization operations can be adapted to produce a sequence of simple operations that generate a generalization chain which can incrementally be sent to a client.

### 5.1 Polyline simplification based on 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 inspect a line and decide upon which point can be omitted. The most popular algorithm is the globally operating algorithm by Douglas and Peucker. In order to decompose the point reduction process into a sequence of reversible elementary generalization operations, a Binary Line Generalization (BLG) tree – a scale dependent decomposition of a line – is generated by recursively extending the levels of detail and describing it in a tree structure (see Figure 1). The root of the tree represents the most coarse line consisting of start and endpoint only; the inner nodes stand for intermediate generalization levels specifying line sectors with an associated generalization level, and the leaf nodes, finally, contain the original line elements – their 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 line sector  $AF$  at scale level  $c$  a split of the line into the two sectors  $AC$  and  $CF$  will occur. In order to present the line in a certain level of detail, the tree has to be traversed down to the node with the given scale level.

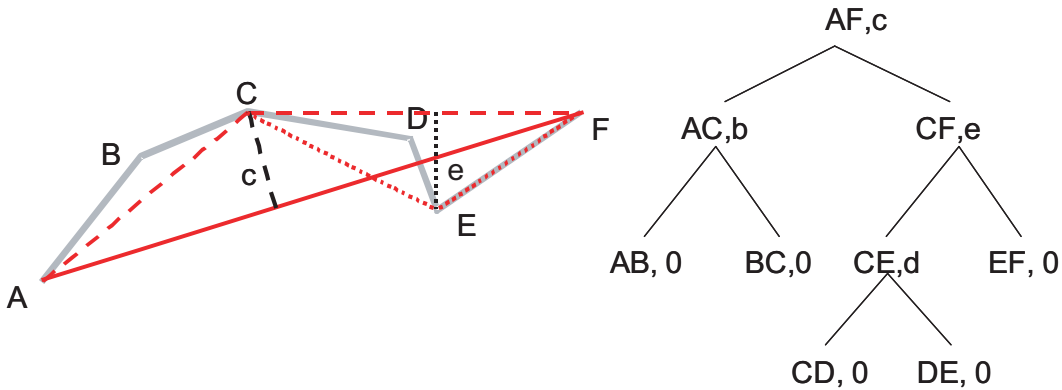


Fig. 1. Original line  $ABCDEF$  in grey with first two levels of Douglas-Peucker-decomposition in red (left) and corresponding BLG-tree (right): nodes represent line segments and corresponding scale level at which child nodes are expanded.

The full BLG-tree has to be generated in order to allow for the full zooming

from coarse to fine. The transformation into the SOs is straightforward (cf. Table 3): Starting point is a new line  $AF$  which appears at a certain scale level  $EPS$ , corresponding to its length  $length(AF)$ . The line is generated by creating vertex  $v_0$  at position  $A$  (`NPR xA yA`), duplicating this vertex (`DV 0`), thus generating vertex  $v_1$  and moving the duplicated vertex to the position  $F$  by increments  $\Delta x_{AF} = x_F - x_a$ ,  $\Delta y_{AF} = y_F - y_A$  (`MV 1 delta_x_AF delta_y_AF`). At scale level  $c$  (`EPS c`) a new vertex is inserted: this is accomplished by duplicating vertex  $A$  (i.e. vertex  $v_0$  in the internal numbering scheme) and moving it to position of point  $C$  by increments  $\Delta x_{AC}$ ,  $\Delta y_{AC}$ . Alternatively, a vertex  $v_1$  could be inserted on edge  $AF$  at the position of the projection of point  $C$  onto line  $AF$  using the `IV`-operation. It can be observed from the table that the numbering of the nodes and lines is continuously adjusted in order to preserve the correct sequence. All required information can be immediately derived from the BLG-tree. The only issue is an appropriate sequencing of the insertion of the points, taking the respective scale levels of the nodes into account.

The necessary simple operations are duplicating (or inserting) and moving vertices, i.e. `DV`, `IV`, `MV`. These operations could be combined to an ‘insert-point’-EGO, taking the edge and the position of the new point as input.

Figure 2 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 a slider. Moving the slider to the right leads to a refinement, i.e. a further traversal of the tree, moving it to the left leads to a coarsening of the representation. Furthermore, the transmission is organized in a way that only data in the current extent will be loaded and refined (for details see Brenner and Sester (2005)).

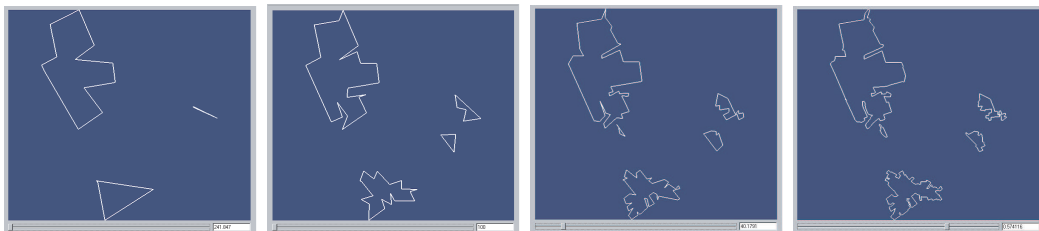


Fig. 2. Visualization of different stages of generalization triggered by different scale values by moving a slider.

## 5.2 Building simplification

Building simplification is a special case of a point reduction method, where the specific properties of buildings are taken into account. Here, the point reduction is more a structure reduction, as properties like parallelism and rectangularity have to be respected by the algorithm. We used a method that

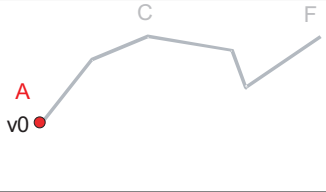

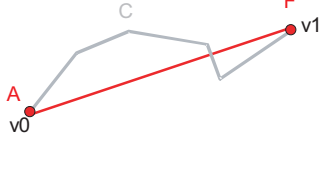
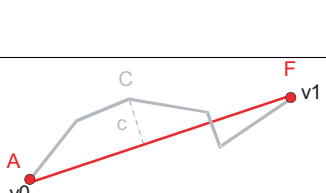
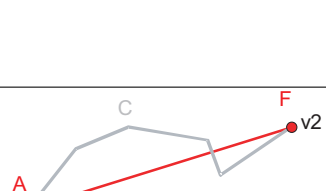
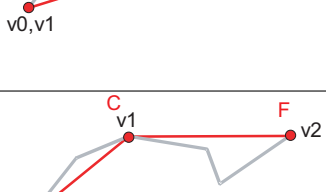
code	explanation	visualization
POLY	create new object	
EPS length(AF)	scale level EPS corresponds to distance between points A and F	
NPR xA yA	create vertex $v_0$ with coordinates $x_A$ and $y_A$	
DV 0	Duplicate vertex $v_0 \rightarrow$ create vertex $v_1$	
MV 1 delta_x_AF delta_y_AF	Move vertex $v_1$ by $\Delta x_{AF}$ and $\Delta y_{AF} \rightarrow$ move it to point F	
EPS c	New scale level is at value EPS=c	
DV 0	Duplicate vertex $v_0 \rightarrow$ create new vertex $v_1$	
MV 1 delta_x_AC delta_y_AC	Move this new vertex $v_1$ by $\Delta x_{AC}$ and $\Delta y_{AC}$ to point C	
...	...	

Table 3

Coding Douglas-Peucker line simplification using the vocabulary of SOs leading to an incremental refinement of the geometry (in red) (see also Figure 1).

analyzes the shape of the building locally 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.

The description of the generalization of these three structures in terms of SOs is straightforward. The generalization of an offset is visualized in Figure 3: An offset consisting of four nodes  $v_1, \dots, v_4$  is replaced by a straight line. The

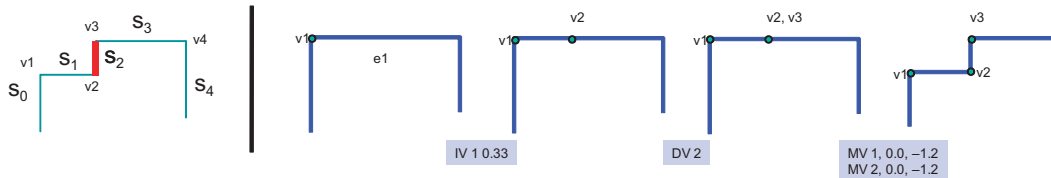


Fig. 3. Sequence of operations to generate an offset: detailed situation (left); inverse generalization operations (four following figures).

generalization process eliminates the edge  $s_2$  that is too short to be represented in the target scale. In order to do so, the longest edge adjacent to the short edge  $s_2$ , in this case it is edge  $s_3$ , is extended. A new point is created at the intersection of the extended edge and the predecessors predecessor edge (in this case between edges  $s_3$  and  $s_0$ ). In order to code the inverse process in terms of SOs, we start from the end situation with an edge  $e_1$ , then insert a vertex on this edge at 33% of the line length, thus creating vertex  $v_2$ . Then this vertex is duplicated, which generates vertex  $v_3$  at the same position. Moving vertices  $v_1$  and  $v_2$  to their final position ends the process. In summary, an EGO for an offset can be defined by combining the four SOs in Figure 3. In a similar way, the generalization operations for the other two scale events extrusion and corner can be coded in terms of EGOs and SOs. In Figure 4, the generalization of four buildings is shown: the snapshots visualize how – at certain stages of the parameter  $\mathbf{EPS}$  that describes the discernable minimal distance  $s_2$  in an object – more and more buildings as well as building details appear.

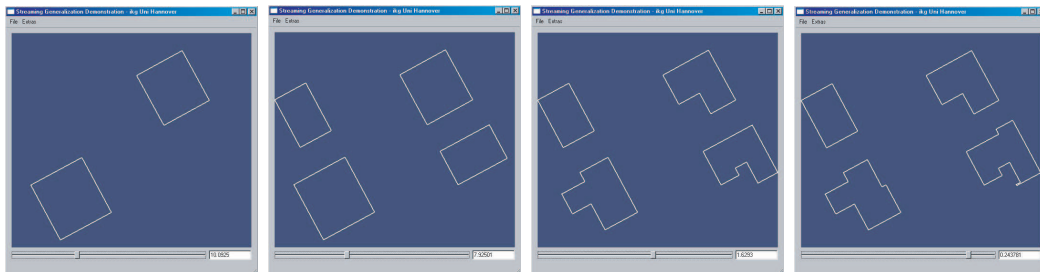


Fig. 4. Simplification of four buildings: Screenshots visualizing increasing refinement (from left to right).

### 5.3 Typification

Typification involves a group of objects that is replaced by a new group with fewer objects. This means that considerable changes occur between the different scales as objects are eliminated and replaced by new ones. Coding this process in terms of EGOs is simple: a set of objects collapses and new objects emerge. The collapse-EGO is described by a set of MV-operations, where the boundary points of the object move to the center point and thus lead to a disappearance of the object. Creating a new geometry is the inverse process: the center point is created by a NPR-command, then this vertex is duplicated  $n - 1$  times to create the  $n$  boundary points, which are then moved to their correct position using the MV-command (see Brenner and Sester (2005) for more details).

### 5.4 Displacement

The coding of the displacement operation in terms of SOs is very simple, as it only consists of move-operations (MV) of the original points to their new displaced positions. We use a least squares adjustment based approach for calculating the displacement between all objects in a scene (Sester (2005)). Figure 5 shows an example for a spatial situation before and after displacement. The algorithm leads to an optimal solution of all spatial conflicts between all objects. The displacement is coded by MV-operations of the individual vertices. These node-displacements are visualized with little red arrows in Figure 5, right. They also clearly show that objects are only shifted in areas where spatial conflicts occur and no object changes are needed elsewhere.

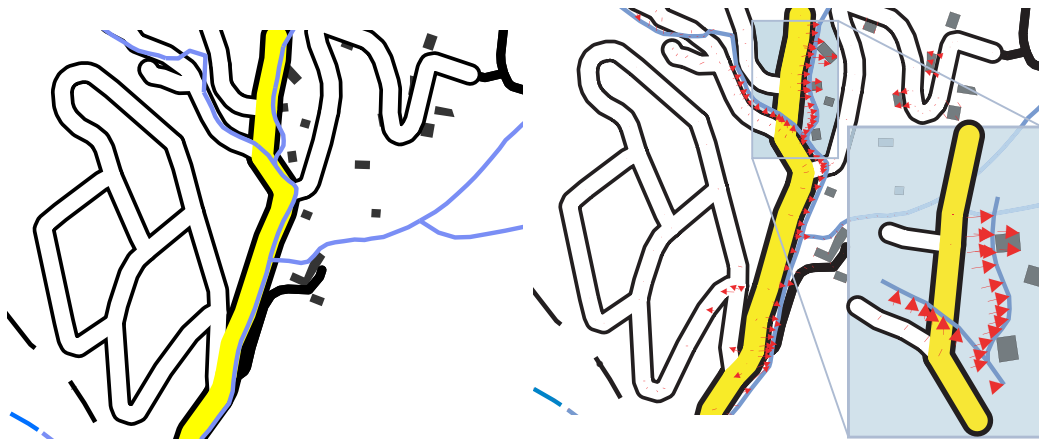


Fig. 5. Displacement: situation before (left) and after displacement (right): the spatial conflicts between all objects, especially rivers and streets, are resolved. Displacement vectors visualize MV-operations.

## 6 Coding efficiency

In order to compare the storage requirements of the coding in terms of SOs 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.

In the course of simplification a line consisting of  $n$  points is reduced to 1 point and then vanishes (or vice versa, for progressive transmission, it comes into existence with 1 point and then iteratively is refined by including new points until its detailed structure is reached). This line is stored with  $n$  double values (for  $x$  and  $y$  coordinate) in the original representation. An incremental generalization of this line is creating additional  $n - 1$  different representations, each of which consists of one point less than the previous representation. Transmitting all the possible  $n$  representations would require

$$n + (n - 1) + \dots + 3 + 2 + 1 = \frac{1}{2}n(n + 1)$$

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  $O(n^2)$ .

Storing this information in terms of SOs requires two operations for each intermediate point (DV <int>, MV <float> <float>), which in turn requires  $n$  points or  $2 * n$  coordinate differences. As the coordinate differences are typically small, float values can be used or even a coding scheme based on integers. In addition to the points, also the operation codes (DV, MV) together with integer values describing the point identifiers 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 cost of transmitting the most detailed one.

Coding *displacement* is more demanding concerning the data volume, as it requires the movement of potentially all  $n$  points. However, firstly, the shift-values are small, as the movements of the points are typically very small compared to the large coordinate values, and thus again can be coded using float values or integers. Secondly, only changes are encoded, thus not the whole data set has to be transferred in all scale-steps (see also displacement vectors in Figure 5 that are not homogeneously distributed in the scene, as they occur only in areas of conflicting objects). Finally, a complex EGO can be defined that encodes the movement of an object as a whole. This does, however, require that there are no object deformations, which cannot always be assumed.

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 when scales change, and hardly a reduction in data volume can be achieved.

## 7 Discussion

### 7.1 *The role of EGOs*

As mentioned earlier, the coding in terms of EGOs allows grouping several SOs to higher level standard operators. In the current implementation this has not yet been done, and only the SOs were used. EGOs promise to be beneficial e.g. for describing the displacement of whole objects and also the typification of object groups efficiently. EGOs allow for a more abstract and compact representation of higher level operations. This would also possibly reduce the volume of the code to be transmitted. Its implementation allows to code arbitrary abstract geometry and topology modifications. In this way generalization operations like ‘extrude-building-part’, ‘insert-rectangle’ or ‘displace-object’ can be realized and implemented. Similarly, for typification, a command can take a group of objects and replace it by another group with fewer objects. A prerequisite is, that the definition of the EGOs in terms of SOs is transmitted to the client prior to sending any data. In this presentation we focused on a proof-of-concept for which an implementation based on the elementary geometric and topologic commands was sufficient.

### 7.2 *Determination of scale values*

In the case of point reduction and building simplification, the object geometry itself defines discrete generalization levels. E.g. a line with 5 points creates 4 scale levels at which intermediate points are inserted. Similarly, a building generalization is determined by the discrete set of the facade points. In this way, *all intermediate representations* of an object are generated and the multiscale coding scheme represents all the natural, inherent scales of an object. Thus, in the case of discrete generalization operations, only discrete steps in the continuum of the scale space need to be explicitly represented, as between them, no further changes can occur.

The continuous generalization operations do not have this nice property: there, in theory, each infinite intermediate scale value could also be relevant. Consider e.g. the enlargement of an object: every scale value leads to a corresponding different size of the object. The same holds for the displacement: every scale

value leads to a dedicated representation of the objects in the scene. So, in the continuous case, intermediate target scales have to be pre-defined, based on which subsequently the generalization is calculated and the multiscale representation is generated. The selection of these scales is dependent on the scale changes that the user is willing to tolerate and the resolution of the display device.

### *7.3 Integration of multiple generalization operations*

Applying the concept of SOs and EGOs for the integration of different operations is straightforward: the coding sequences for the individual objects can be just concatenated. The system reads the SOs and evaluates them depending on the corresponding EPS values. As a matter of fact, it has to be assumed that the generalization processes do not influence each other, e.g. the simplification of roads and the simplification of buildings do not affect each other. This assumption is justified at least in large scale representations. Otherwise, integrated generalization processes have to be devised that generate the SOs and the EGOs.

## **8 Continuous Visualization**

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 SOs and EGOs 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 operation. For example, the of a building part (see Figure 6) would be interpreted as 'move extrusion in small steps until full extent is reached'. We term this approach continuous visualization 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.

In the current implementation the changes are visualized one after the other. With the animation of geometric changes, a smooth transition from one representation to the next is possible. However, for operations like typification, where the changes between two representations can be dramatic, other visualization schemes might be more appropriate. van Kreveld (2001) proposes to blend previous and subsequent representation for a short time, before the old



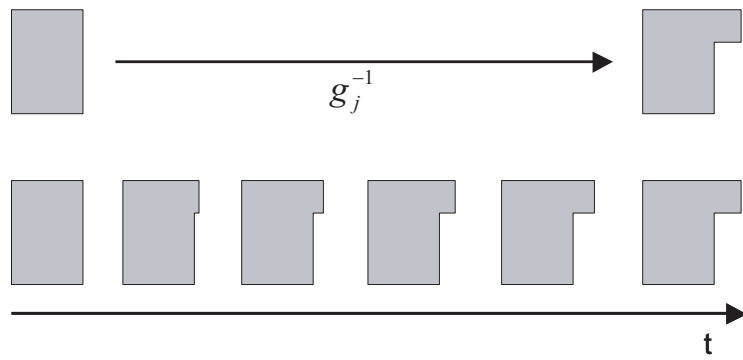


Fig. 6. Animation of discrete geometric change: without animation: popping effect (top), interpolating a series of intermediate positions (bottom).

one is faded out. This can easily be achieved with the setting of the EPS-parameters which control the appearance and disappearance of the objects.

With typification e.g. 16 buildings can be replaced by four and subsequently by a single building, before it finally disappears in the smallest scale. This can be realized in terms of SOs as described in the previous section.

In order to realize a visual blending of these discrete changes when zooming in, the more detailed objects are created before the coarser objects disappear. Figure 7 shows how the transitions between the discrete steps a), c) and e) are visually enhanced by blending both representations for a small scale range. In this way, e.g. the coarse and intermediate representations a) and c) are both visible in the scale range between  $\text{EPS}=10$  and  $\text{EPS}=8$  as shown in b). This visual blending could even be enhanced by also smoothly fading the color.

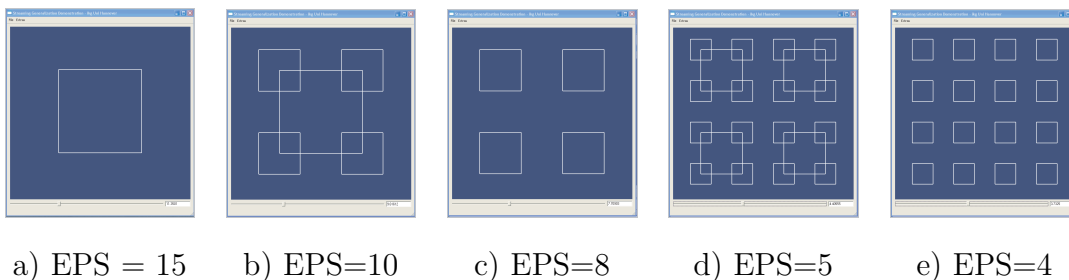


Fig. 7. Blending of different generalization levels.

## 9 Summary and outlook

An approach was presented to decompose changes in object geometry into a small set of simple operations. These operations express the creation of objects as well as the iterative refinement of their shapes. This coding scheme was used to represent different generalization levels of objects efficiently. It was shown how this representation can be generated by different generalization operations. As compared to the naive approach of storage and transmission of  $n$  different geometric representations, requiring  $O(n^2)$  transmission space, coding of the differences needs only  $O(n)$  for the case of point reduction. For other generalization operations the coding efficiency depends on the degree of changes occurring in the different scale steps. Besides incrementally presenting the changes in the geometry, it was also shown that the changes can be animated, leading to nearly invisible changes between the different representations when going from one scale to the next.

In the current approach the generation of the SOs is an off-line pre-processing step. It is possible to generate the code on-the-fly upon request by the client.

Then, however, the response time on the client will be additionally delayed by the time needed by the generalization process. Thus, the trade-off between additional storage of the code and processing time has to be balanced.

Aggregating SOs to higher level EGOs promises to be beneficial for some kinds of generalization operations, e.g. the creation, removal or movement of whole objects (e.g. typification, displacement) or the structured modification of object shapes (e.g. inserting or removing offsets of buildings). However, for the proof of concept presented in this paper, it was sufficient to show the functionality and flexibility of the SOs.

Future work will investigate possible extensions of the approach to three dimensions, especially the definition of the required elementary operations.

ACKNOWLEDGEMENT: This research is partially funded by the VolkswagenStiftung. We thank the anonymous reviewers for their valuable suggestions.

## References

- Ai, T., Li, Z., Liu, Y., 2004. Progressive transmission of vector data based on changes accumulation model. In: Fisher, P. (Ed.), *Developments in Spatial Data Handling - 11th International Symposium on Spatial Data Handling*. Springer, pp. 86–96.
- Bertolotto, M., Egenhofer, M., 1999. Progressive vector transmission. In: *Transactions of the ACMGIS99*. Kansas City, MO, pp. 152–157.
- Brenner, C., Sester, M., 2005. Continuous generalization for small mobile displays. In: Agouris, P., Croitoru, A. (Eds.), *Next Generation Geospatial Information*. ISPRS Book Series, Taylor and Francis Group, pp. 33–41.
- Cecconi, A., Weibel, R., Barrault, M., 2002. Improving automated generalisation for on-demand web mapping by multiscale databases. In: *Symposium on Geospatial Theory, Processing and Applications – Spatial Data Handling*. ISPRS, IGU, CIG, Ottawa, Canada, pp. 515–532, cD-Rom.
- 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.
- Harrie, L., 2001. An optimisation approach to cartographic generalization. Ph.D. thesis, Department of Technology and Society, Lund Institute of Technology, Lund University, Sweden.
- Højholt, P., 1998. Solving local and global space conflicts in map generalization using a finite element method adapted from structural mechanics. In: Poiker, T., Chrisman, N. (Eds.), *Proceedings of the 8th International Symposium on Spatial Data handling*. Vancouver, Canada, pp. 679–689.
- Hoppe, H., 1996. Progressive meshes. In: *Proceedings of SIGGRAPH 96*, ACM

- SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series. New Orleans, LA, pp. 99 – 108.
- Mackanness, W., Ruas, A., Sarjakoski, L. (Eds.), 2007. Generalisation of Geographic Information: Cartographic Modelling and Applications. Elsevier, published on behalf of the International Cartographic Association.
- Müller, J., Wang, Z., 1992. Area-patch generalisation: a competitive approach. *The Cartographic Journal* 29, 137–144.
- Nöllenburg, M., Merrick, D., Wolff, A., Benkert, M., 2008. Morphing polylines: A step towards continuous generalization. *Computers, Environment and Urban Systems* 32, 248–260.
- Regnault, N., 1996. Recognition of building clusters for generalization. In: Kraak, M., Molenaar, M. (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.
- Regnault, N., McMaster, R., 2007. Generalisation of Geographic Information: Cartographic Modelling and Applications. Elsevier, published on behalf of the International Cartographic Association, Ch. 3: A synoptic view of generalization operators, pp. 37–66.
- Sarjakoski, T., Sarjakoski, L., Lehto, L., Sester, M., Illert, A., Nissen, F., Rystedt, R., Ruotsalainen, R., August 2002. Geospatial info-mobility services - a challenge for national mapping agencies. In: *Commission IV Symposium on Geospatial Theory, Processing and Applications*. Vol. 34/4. International Society of Photogrammetry and Remote Sensing, Ottawa, Canada, p. 5, cD-Rom.
- Sester, M., 2000. Generalization based on least squares adjustment. In: *International Archives of Photogrammetry and Remote Sensing*. Vol. 33. International Society of Photogrammetry and Remote Sensing, Amsterdam, Holland, pp. 931–938.
- Sester, M., 2005. Optimizing approaches for generalization and data abstraction. *International Journal of Geographical Information Science* 19 (8-9), 871–897.
- Sester, M., 2007. Self-organizing maps for density-preserving reduction of objects in cartographic generalization. In: Agarwal, P., Skupin, A. (Eds.), *Self-Organising Maps. Applications in GI Science*. Wiley John and Sons, pp. 107–120.
- Thiemann, F., 2002. Generalization of 3d building data. In: *Commission IV Symposium on Geospatial Theory, Processing and Applications*. Vol. 34/4. International Society of Photogrammetry and Remote Sensing, Ottawa, Canada, cD-Rom.
- van Kreveld, M., 2001. Smooth generalization for continuous zooming. In: *Proceedings of the 20th International Cartographic Conference*. Beijing, China, pp. 2178–2185.
- van Oosterom, P., 1995. The gap-tree, an approach to 'on-the-fly' map generalization of an area partitioning. In: Müller, J.-C., Lagrange, J.-P., Weibel, R. (Eds.), *GIS and Generalization - Methodology and Practice*. Taylor &

- Francis, pp. 120–132.
- van Oosterom, P., 2005. Variable-scale topological data structures suitable for progressive data transfer: The gap-face free and gap-edge forest. *Cartography and Geographic Information Science* 32 (4), 331–346.
- Yang, B., Purves, R., Weibel, R., 2007. Efficient transmission of vector data over the internet. *International Journal of Geographical Information Science* 21 (1-2), 215–237.