International Journal of Geographical Information Science, 2014
Vol. 28, No. 3, 455–478, http://dx.doi.org/10.1080/13658816.2013.809724

Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data

Peter van Oosterom* and Martijn Meijers

Department of GIS Technology, OTB Research Institute for the Built Environment, Delft University of Technology, Delft, The Netherlands

(*Received 22 August 2012; final version received 20 May 2013*)

This paper introduces the concept of the smooth topological Generalized Area Partitioning (tGAP) structure represented by a space-scale partition, which we term the space-scale cube. We take the view of 'map generalization as extrusion of data into an additional dimension'. For 2D objects the resulting vario-scale representation is a 3D structure, while for 3D objects the result is a 4D structure.

This paper provides insights in: (1) creating valid data for the cube and proof that this is always possible for the implemented 2D tGAP generalization operators (line simplification, merge and split/collapse), (2) obtaining a valid 2D polygonal map representation at arbitrary scale from the cube, (3) using the vario-scale structure to provide smooth zoom and progressive transfer between server and client, (4) exploring which other possibilities the cube brings for obtaining maps having non-homogenous scales over their domain (which we term mixed-scale maps), and (5) using the same principles also for higher dimensional data; illustrated with 3D input data represented in a 4D hypercube.

The proposed new structure has very significant advantages over existing multi-scale/multi-representation solutions (in addition to being truly vario-scale): (1) due to tight integration of space and scale, there is guaranteed consistency between scales, (2) it is relatively easy to implement smooth zoom, and (3) compact, object-oriented encoding is provided for a complete scale range.

Keywords: 3D modelling; 3D visualization; generalization; geographic information systems; Internet GIS

1. Introduction

Technological advancements have led to maps being used virtually everywhere; take, for example, their use in mobile smartphones. Map use is more interactive than ever before: users can zoom in, out and navigate on the (interactive) maps. Therefore recent map generalization research shows a move towards continuous generalization. Despite some useful efforts (van Kreveld 2001, Cecconi and Galanda 2002, Sester and Brenner 2005), there is no optimal solution yet.

This paper introduces the first true vario-scale structure for geographic information: a small step in the scale dimension leads to a small change in representation of geographic features that are represented on the map. Continuous generalizations of real-world features can be derived from the structure that can be used for presenting a smooth-zoom action to the user. Furthermore, mixed-scale visualizations can be derived with more and less generalized features shown together in which the objects are consistent with each other.

*Corresponding author. Email: p.j.m.vanoosterom@tudelft.nl

Making such a transition area is one of the principal difficulties for 3D computer graphic solutions (e.g. using stitch strips based on triangles Noguera *et al.* 2010).

The remainder of this paper is structured as follows: Section 2 contains a discussion how the classic topological Generalized Area Partitioning (tGAP) structure can be represented by a 3D space-scale cube (SSC) and how this can be adapted to store a more continuous generalization. The use and application of the smooth tGAP structure is further discussed in Section 3. Section 4 proves that for all configurations it is possible to create the smooth tGAP structure: including convex areas and areas with holes. The presented data structures and methods are valid for both 2D and 3D data. Section 5 discusses how the method is used for 3D data resulting in a 4D SSC representing the smooth tGAP structure. The paper is concluded with a short description of the main results, together with a summation of a long list of open research questions, in Section 6.

2. The smooth tGAP structure

The tGAP structure has been presented as a vario-scale structure (van Oosterom 2005). In summary, the tGAP structure traditionally starts with a planar partition at the most detailed level (largest scale). Next the least important object (based on geometry and classification) is selected, and then merged with the most compatible neighbour (again based on geometry and classification). This is repeated until only a single object is remaining. The merging of objects is recorded in tGAP-tree structure and the last object is the top of the tree. The (parallel) simplification of the boundaries is also executed during this process and can be recorded in a specific structure per boundary: the BLG-tree (binary line generalization). As assigning the least important object in certain cases to just a single neighbour may result in a suboptimal map representation, the weighted split, assigning the various parts to multiple neighbours was introduced. This changed the tGAP-tree into a tGAP directed acyclic graph (DAG) and together with the BLG-tree, this is called the tGAP structure. The tGAP structure can be seen as result of the generalization process and can be used efficiently to select a representation at any required level of detail (LoD; scale or height). Figure 1 shows four map fragments and the tGAP structure in which the following generalization operations have been applied:

(1) Collapse road object from area to line (split area of road and assign parts to the neighbours);
(2) Remove forest area and merge free space into neighbour farmland;
(3) Simplify boundary between farmland and water area.

2.1. The tGAP structure represented by the 3D SSC

The tGAP structure is a DAG and not a tree structure, as an object can have several parent objects at a less detailed level. For example, it can be seen that the split causes the road object to have several parents; see Figure 1e. In our current implementation the simplify operation on the relevant boundaries is combined with the remove or collapse/split operators and is not a separate step. However, for the purpose of this paper it is more clear to illustrate these operators separately. For the tGAP structure, the scale has been depicted as third dimension – the integrated SSC representation (Vermeij *et al.* 2003, Meijers and van Oosterom 2011). We termed this representation the SSC in analogy with the space-time cube as first introduced by Hägerstrand (1970). Figure 2a shows this representation for the example scene of Figure 1. In the the vario-scale 2D area objects

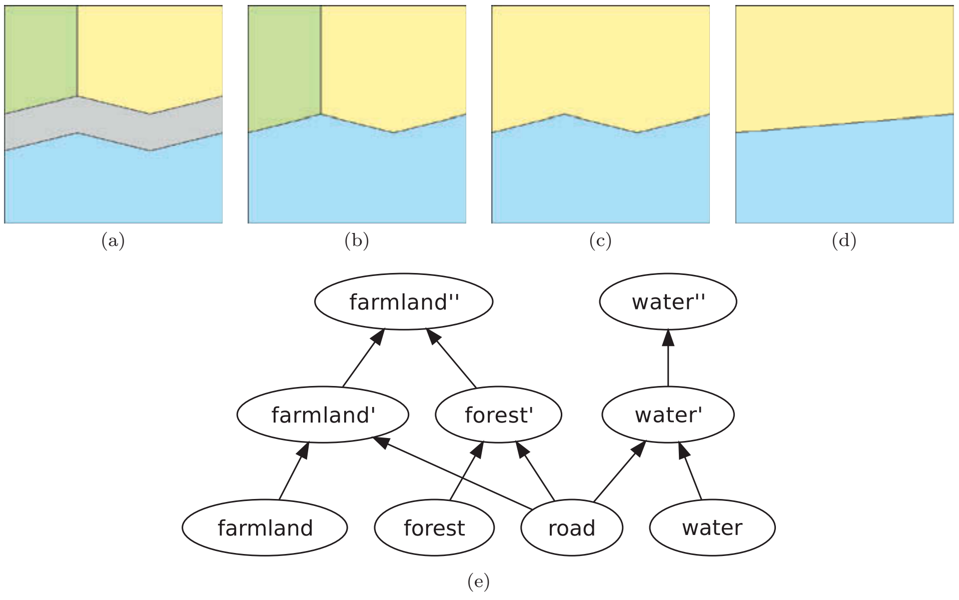

Figure 1. The four map fragments and corresponding tGAP structure: (a) original map, (b) result of collapse, (c) result of merge, (d) result of simplify, (e) corresponding tGAP structure.

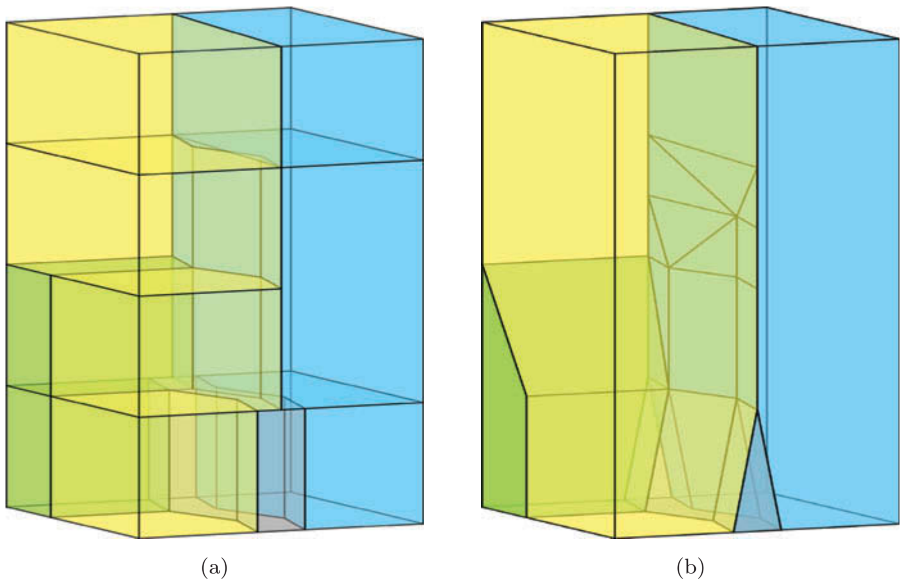

Figure 2. The space-scale cube (SSC) representation in 3D: (a) SSC for the *classic* tGAP structure, resulting in a collection of stacked prisms, (b) SSC for the *smooth* tGAP structure, resulting in a collection of arbitrarily shaped polyhedra.

are represented by 3D volumes (prisms), the vario-scale 1D line objects are represented by 2D vertical faces (for example, the collapsed road) and the vario-scale 0D point object would be represented by a 1D vertical line. Note that in the case of the road area collapsed

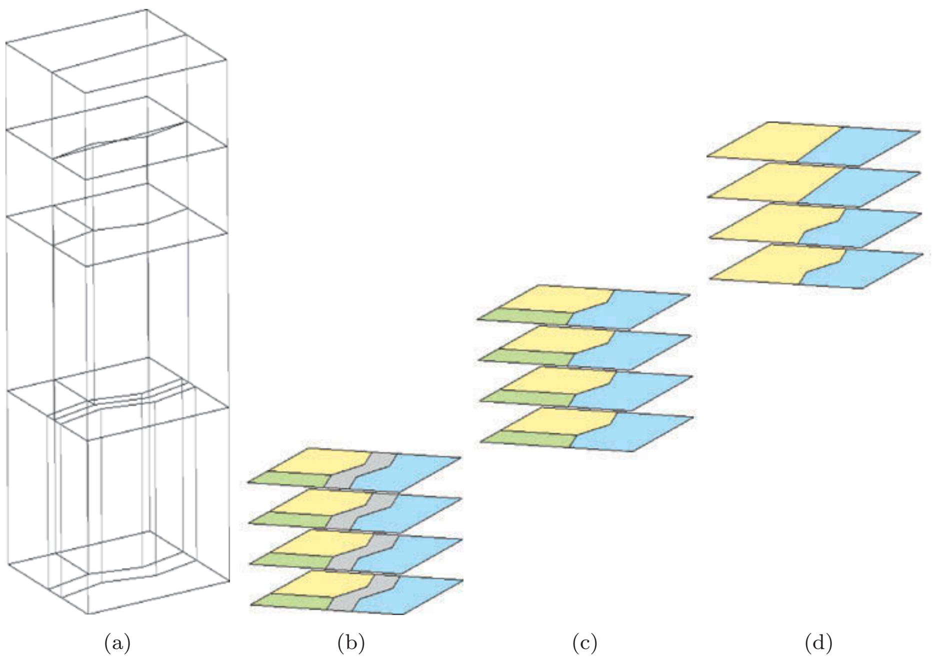

Figure 3. The map slices of the classic tGAP structure: (a) wireframe of (classic) space-scale cube, (b) step 1 (collapse), (c) step 2 (merge) and (d) step 3 (simplify). Note that nothing changes until a true tGAP event has happened.

to a line, the vario-scale representation consist of a compound geometry with a 3D volume-part and 2D face-part attached.

Though many small steps (from most detailed to most coarse representation – in the classic tGAP, $n - 1$ steps exist, if the base map contains n objects), this could still be considered as many discrete generalization actions approaching vario-scale, but not *true* vario-scale. Split and merge operations cause a sudden local ‘shock’: a small scale change results in a not so small geometry change where, for example, complete objects disappear; see Figure 3. In the SSC this is represented by a horizontal face; a sudden end or start of the corresponding object. Furthermore, polygon boundaries define faces that are all vertical in the cube, i.e. the geometry does not change at all within the corresponding scale range (resulting in prisms constituting a full partition of the SSC).

2.2. Smooth line simplification

In order to obtain more gradual changes when zooming, i.e. in a morphing style (cf. Sester and Brenner 2005, Nöllenburg *et al.* 2008), we first realized that the line simplification operation could also output non-vertical faces for the SSC and that this has a more true vario-scale character; e.g. when replacing two neighbouring line segments by a single new line segment (omitting the shared node), this can be represented by three triangular faces in the SSC; see Figure 4. Note that both the sudden-change line simplification and the gradual-change line simplification have three faces in the SSC: sudden-change has two rectangles and one triangle and gradual-change has three triangles. When slicing a map based on the SSC fragment as depicted in Figure 4b taking a cross-section of the cube at a

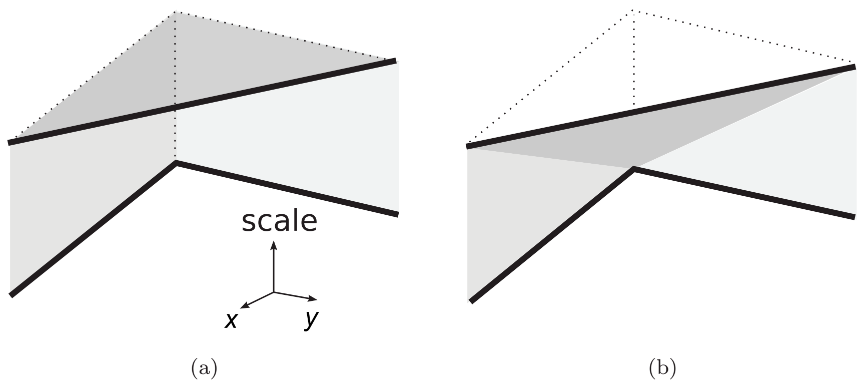

Figure 4. Line simplification in the SSC: (a) sudden removal of node: two rectangles and one triangle, (b) gradual change: three triangles. The dashed lines in (b) only illustrate the difference with the sudden-change variant.

certain scale, a small change in scale leads to a small change in the geometry of the depicted map objects. Note that when slicing at intermediate level the example of Figure 4 will result in a line with four points instead of three at the start (and two at the end). It may feel counter-intuitive that the number of points is temporarily increased during this operation, but this is the 'price' needed for the smooth transition and the final result is a line with fewer points. Furthermore, the more general line simplification (removing more than one node of a polyline) can be considered to consist of several smaller sub-steps: one step for the removal of each of the nodes; see Figure 5.

2.3. Smooth merge and split

The merge and split (collapse) operations can, like the gradual line simplification operation as sketched above, be redefined as gradual actions supporting smooth zoom. For example, in case of the merge of two objects: one object gradually grows and the other shrinks – in a SSC this corresponds to non-vertical faces and there is no more need for a horizontal face, i.e. a suddenly disappearing feature; see Figure 2b. All horizontal faces in the cube are now gone, except the bottom and top faces of the cube. Note that adjacent faces in the same plane belonging to the same object are merged into one larger face, the big front-right face in Figure 2b corresponds to four faces in Figure 2a. The same is true for the involved edges, several smaller edges on straight lines are merged, and the shared nodes are removed. This can be done because they carry no extra information.

Perhaps the most important and elegant consequence of the merging of the different polyhedral volumes is that the number of volumes is reduced: there is a one-to-one correspondence between a single object and its smooth tGAP polyhedral representation, ranging over all relevant map scales. Also, this gives a much more elegant way of attaching attributes (of a single real-world object) to a varied representation. The benefit of a smaller number of primitives, the nodes, edges, faces and volumes, is that there are also fewer topology references needed to represent the whole structure. In previous investigations it was reported that the storage requirements for an explicit topology structure may be as high as, or even higher, than the storage requirements for plain geometry (see previous tests, described in Baars *et al.* 2004, Penninga 2004). This is even

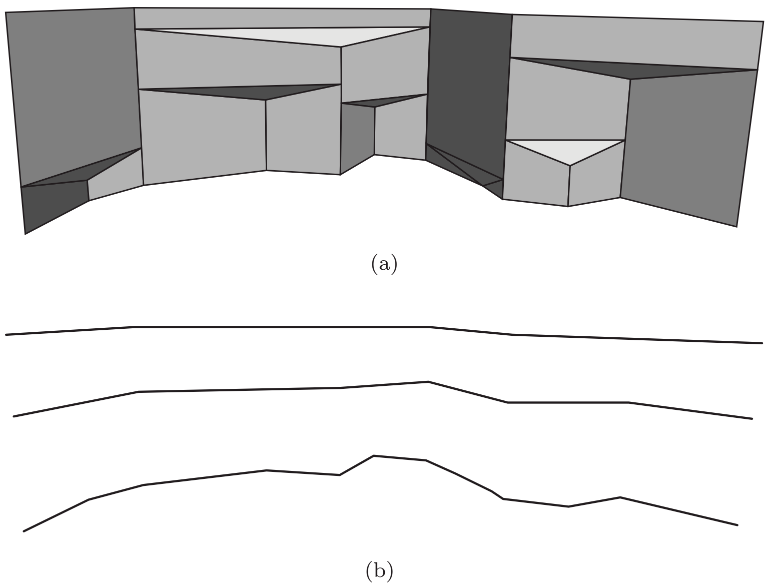

Figure 5. Simplification of an edge and the corresponding space-scale surface visualized in 3D: (a) 3D surface that represents a boundary over multiple scales: note that the dark grey triangles 'lean' forward and the lighter grey triangles 'lean' backward, (b) derived 2D polylines (simplified with the gradual-change line simplification) at multiple scales.

more true for topology based vario-scale data structures (cf. Meijers *et al.* 2009). Lighter structures are more suitable for progressive data transfer and better performance.

Figure 6 illustrates the resulting true vario-scale structure: small deltas in scale will give small deltas for map areas. Figure 7 shows that if all slices of the classic tGAP and the smooth tGAP SSCs are compared, the differences and the benefits of the latter become clear. Note that slicing the SSC at a scale level just before an object is disappearing will result in a thin sliver polygon. This is not considered a problem for dynamic use, but should not be visible at a static 'stop' scale. So, if this would occur, then the requested scale should be adjusted to the nearest appropriate map scale; e.g. a level at which there exists a node in the SSC.

3. Advanced use of the smooth tGAP

This section discusses a number of more advanced usages of the smooth tGAP structure. Subsection 3.1 illustrates how a mixed-scale representation can be obtained from the smooth tGAP structure. Subsection 3.2 explains how the structure can be used to support progressive transfer in a client–server setting.

3.1. Mixed-scale representations

So far, only horizontal slices were discussed for creating 2D maps with homogenous scale, however nothing prevents us from taking *non-horizontal* slices. Figure 8 illustrates a mixed-scale map derived as a non-horizontal slice from the SSC. What does such a non-horizontal slice mean? More detail at the side where the slice is close to the bottom of the

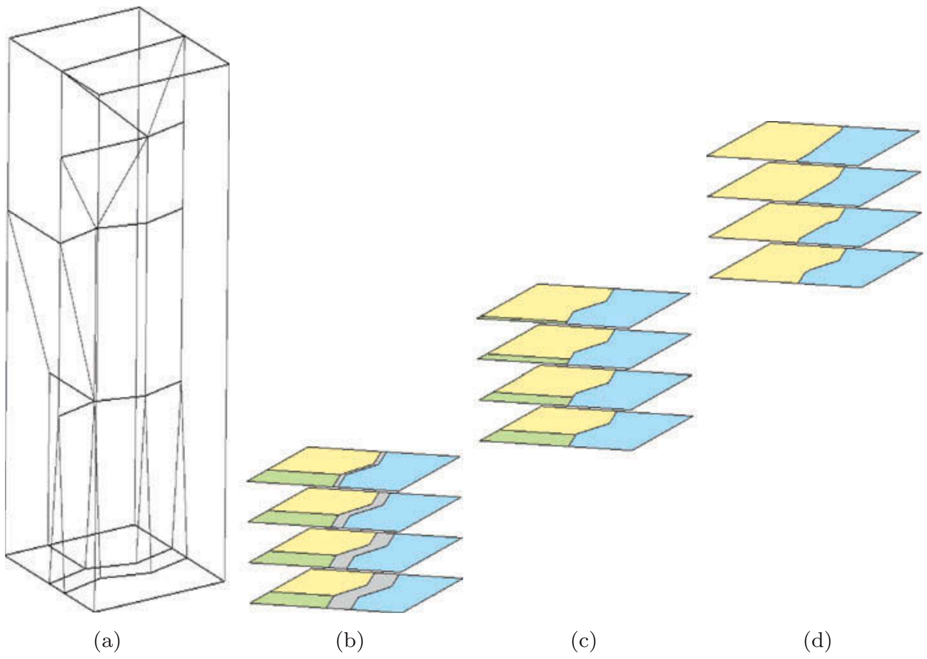

Figure 6. The map slices of the smooth tGAP structure: (a) wireframe of (smooth) space-scale cube, (b) step 1 (collapse), (c) step 2 (merge), (d) step 3 (simplify). Note the continuous changes, also in between the 'true' tGAP events.

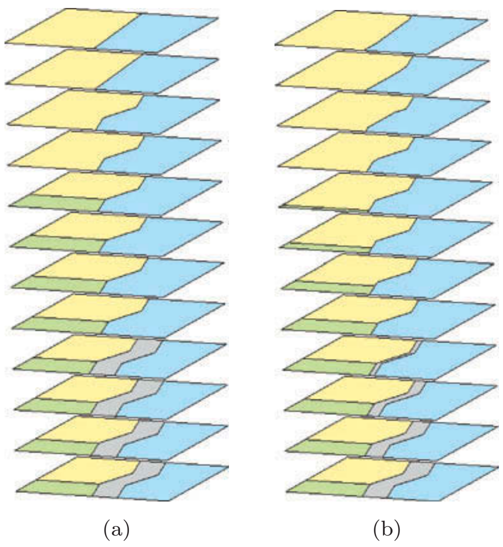

Figure 7. The maps – slices of (a) the classic and (b) the smooth tGAP structure – compared. Note that the stacks of slices correspond to the stacks from Figures 3 and 6. By showing them side by side it is easier to compare the differences of the resulting slices.

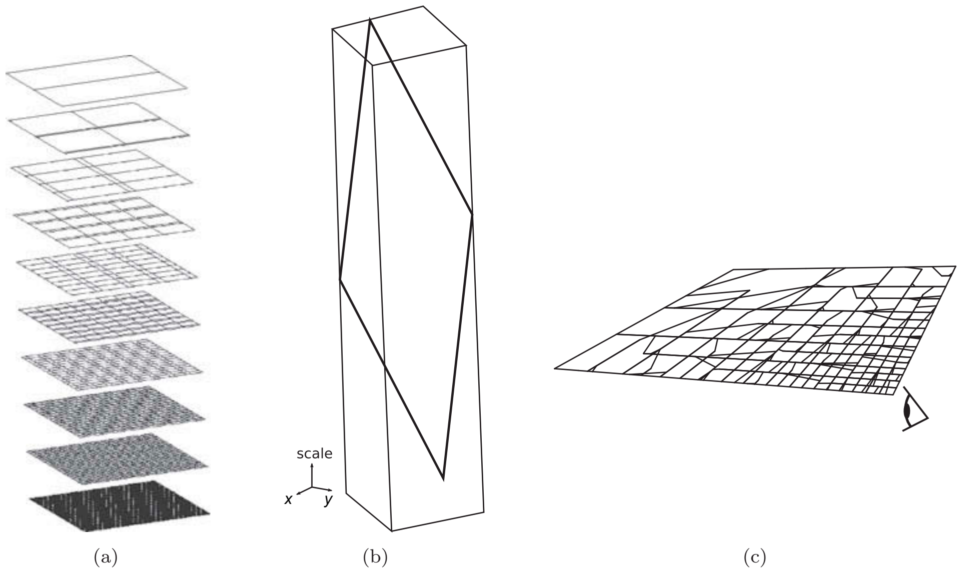

Figure 8. Checkerboard data as input: each rectangular feature is smoothly merged to a neighbour. Note that all merge operations have been executed in parallel, see Subsection 4.1. (a) A stack of horizontal slices, (b) taking a non-horizontal slice leads to a 'mixed-scale' map, (c) one mixed-scale slice (non-horizontal plane), shown in perspective.

cube, less detail where the slice is closer to the top. Consider 3D visualizations, where close to the eye of the observer lots of detail is needed, while further away not so much detail. Such a slice leads to a *mixed-scale* map, as the map contains more generalized (small scale) features far away and less generalized (large scale) features close to the observer. Note that if the slice plane has the exact same angle as one of the object faces, then a delta slice plane movement could result in a sudden big map change: a complete object disappearing at once. As this is an exotic use case and unlikely that exact same angles will occur, this drawback is not really considered a problem.

The mixed-scale representation can also be obtained by slicing surfaces that are non-planar; e.g. a bell-shape surface that could be used to create a meaningful 'fish-eye' type of visualizations (see Figure 9). This is likely to be the hoped-for result in most cases, but it might be true that a single area object in the original data set might result in multiple parts in the slice (but no overlaps or gaps will occur in the slice). What are other useful slicing surface shapes? A folded surface seems to be non-sensical as it could lead to two representations of the same object in one map/ visualization.

The artificial checkerboard 'map' is a kind of worst case example as the map reader would expect to see only rectangular shapes in the output. The combination of smooth transitions (tilted faces) which are intersected by the mixed scale diagonal slicing plane results in some non-orthogonal shapes. By contrast the (non-smooth) classic tGAP encoded in SSC (as in Figure 2a) for generating the mixed-scale output will only generate rectangular shapes (and in that sense looks more natural). However, this classic tGAP does not support (well) true smooth zooming. More usability tests will be needed to evaluate the (dynamic) map types the users prefer with more realistic map data (see Subsection 6.2).

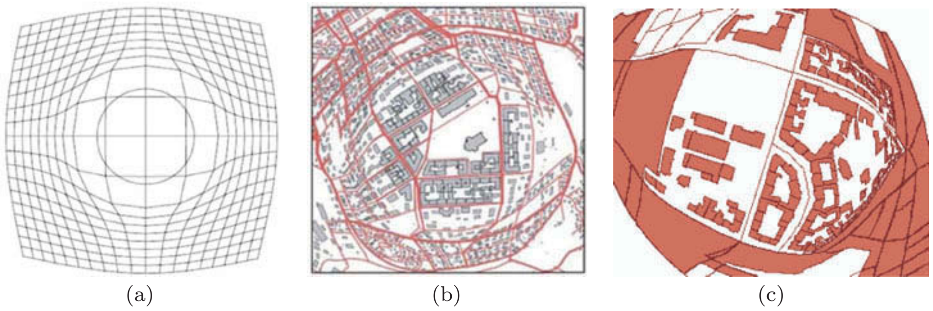
(a) (b) (c)

Figure 9. A ‘mixed-scale’ map. Harrie *et al*. term this type of map a ‘vario-scale’ map, while we term this a ‘mixed-scale’ map. Furthermore, it is clear that there is a need for extra generalization closer to the borders of the map, which is not applied in (b) but is applied in (c). With our solution, this generalization would be automatically applied by taking the corresponding slice (bell-shaped, curved suface) from the SSC. Illustrations (a) and (b) taken from Harrie *et al.* (2002) and (c) from Hampe *et al.* (2004).

3.2. Smooth zoom and progressive transfer

In an online usage scenario where a 2D map is retrieved from the tGAP structures, the amount of vector information to be processed has an impact on the processing time for display on the client. Therefore, as a rule of thumb, we strive to show a fixed number of (area) objects on the map, independent from the LoD the objects have, in such a way that optimal information density is achieved. This number is termed here the *optimal number* of map objects and will be used for retrieving data in such a way that the amount of data to be retrieved on *average* remains constant per viewport (independent from which LoD is retrieved) and thereby the transfer and processing times stay within limits.

The optimal number can be realized because the generalization procedures that create tGAP data lead to progressively less data in the hierarchy, i.e. less data is stored near the top of the SSC where the extent of area objects is considerably larger than at the bottom. A slice (cross-section) in this cube leads to a 2D map. The extent of the viewport (i.e. the window through which the user is looking at the data) also implies that it is necessary clip data out of such a slice: when a user is zoomed out, the viewport of a user will lead to a big geographic extent (the area to be clipped is large) and when a user is zoomed in this extent will become considerably smaller. For a user that performs a panning action it is necessary to move the extent of the clip within the horizontal slicing plane. Figure 10 gives an illustration. A smooth tGAP based server can be arranged to respond to the following types of requests from a smooth tGAP-aware client (illustrated for 2D maps represented by a 3D SSC):

(1) A request to provide an initial map based on simple spatial range overlap selection of the relevant 3D polyhedra representing the vario-scale 2D objects in the requested area A_1 for the requested scale s_1 as illustrated in Figure 10a. Note that the number of selected objects may be relatively large, so it can take some time before a map covering a requested area A_1 can be seen by the client.
(2) A request to provide an initial map with progressive transfer of data (coarse to fine) is based on overlap with a 3D block orthogonal to the axes. The server sends the selected 3D polyhedra smallest scale first (Figure 10b). The client can quickly start drawing an initial course representation, while still receiving additional detail.

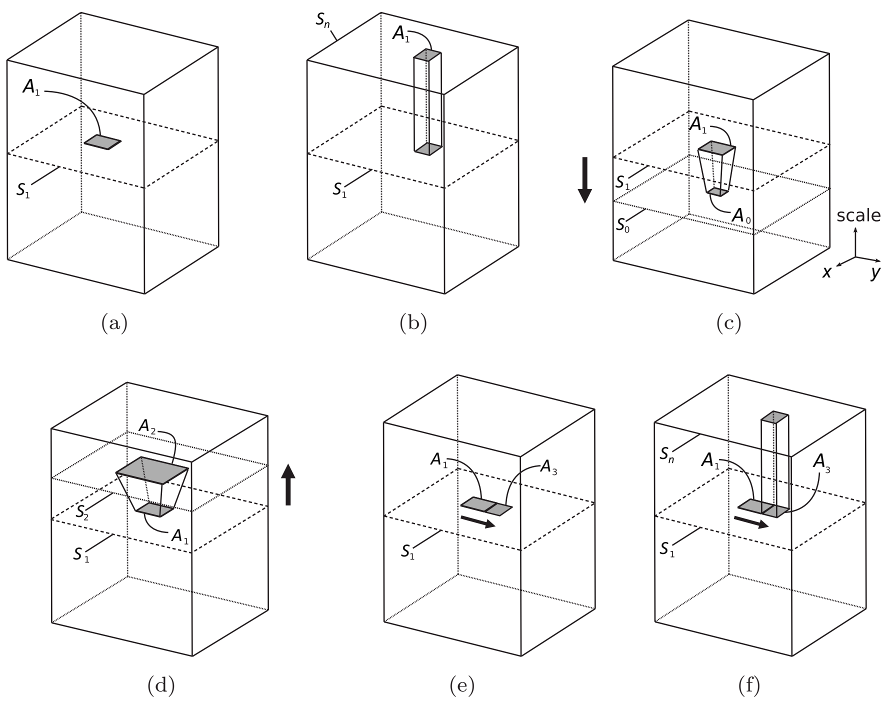

Figure 10. Zooming and panning explained for vario-scale data with the SSC: (a) initial request (non-progressive), (b) initial request (progressive), (c) smooth zooming in (progressive), (d) smooth zooming out (progressive), (e) panning (non-progressive), (f) panning (progressive).

(3) A request to provide the 3D polyhedra for a progressive zoom-in as shown in Figure 10c. Note the shrinking of the spatial selection range from an area A_1 at scale s_1 to an area A_0 at scale s_0 (a truncated pyramid upside down). Alternatively it is possible to provide data for a simple zoom-in. In that case the client does not need to receive ‘intermediate’ 3D polyhedra (this alternative is not depicted in Figure 10).

(4) A request to provide the 3D polyhedra for a progressive zoom-out as shown in Figure 10d. Note the growing of the spatial selection range from an area A_1 at scale s_1 to an area A_2 at scale s_2. In this case the polyhedra are sorted based on largest scale value from the larger to the smaller scale value (which is the reverse order of a zoom-in). Alternatively, also a simple zoom-out is possible without sending ‘intermediate’ polyhedra (again, not depicted in Figure 10).

(5) A request to provide the 3D polyhedra for a simple pan operation from a first area A_1 to an adjacent area A_3 represented at the same scale s_1 as shown in Figure 10e. In that case the server immediately transmits the object data for the new map at the required LoD.

(6) A request to provide the 3D polyhedra for a progressive transfer pan as shown in Figure 10f from a first area A_1 to a second area A_3. In that case the server subsequently transmits more and more detailed data (gradually changing from scale s_n to scale s_1) for the requested spatial range A_3, and the client can gradually increase the LoD with which the image data in said spatial range A_3 is displayed.

Note that the client has to be smooth tGAP-aware, after receiving the polyhedra (in sorted order) it has to perform slicing before the actual display on the screen takes place. In case of 'smooth' zoom-operations, the slicing operations are repeated (at slightly different scale levels) before every display action. Note that an efficient implementation may exploit the fact that the slicing plane is moving monotonically in certain direction (up or down) and may avoid repeating parts of the needed geometric computations. This is similar to the plane-sweep algorithm as used in computational geometry.

Positioning the slice in the cube, together with taking the clip, should lead to an approximately constant number of objects to be visualized. To realize the position of the cross-section means that the question to be answered is 'which height corresponds to the map scale at the client?' In practice this will mean that a thin client will only have to report the current extent (plus its device characteristics) to a server and then can be sure to receive the right amount of data for a specific LoD as the server can translate this extent to a suitable height value to query the data structures. Note that this on *average* is the right amount of information, as there may be regions with more or with less dense content; e.g. rural versus urban area. For a thick client, that has more capabilities and where it is possible to perform more advanced processing, it should be possible to first receive a coarse map, and then to incrementally receive additional details when a user waits longer, leading to a more detailed map (with additional map objects). The mapping of map scale to height in the SSC in this scenario is necessary to determine when to stop sending additional details. Independent from whether a thin or thick client is used: it is key to know what the height value is that is implied by the current map scale of the client (i.e. positioning the height of the slice).

The filled tGAP data structure will be able to supply data for a specific scale range. As we will see, this range is dependent on the optimal number of objects we want to show together with device characteristics.

4. Creating the structure and proof of correctness

This section discusses more aspects of creating the presented smooth tGAP. Two potential issues are presented in the subsections below: (1) can multiple generalization operations be performed in parallel? (Subsection 4.1), and (2) can the prism resulting split and merge operations with horizontal and vertical faces always be transformed into their smooth counterparts with non-horizontal faces? (Subsection 4.2). We provide the (intuitive) proof that the smooth tGAP structure can be created in all situations: including non-convex areas and areas with holes.

4.1. Parallel execution, instead of one by one

Despite the fact that the proposed solution results in a true vario-scale structure, it has still an (old) tGAP drawback and that is the sequencing of all generalization operations. This has the positive effect that it can be proven that all operations are validly represented in the SSC: both at start and end scale of an operation, but also in between. But all gradual changes are local when looking at the big picture: first one operation is (gradually) finished and then the next local operation is started, and so on. This might give a suboptimal smooth-zoom effect – more experiments with end-users are needed to verify whether this is indeed suboptimal.

A solution for the one-by-one sequencing is not implementing the steps in the structure in a sequential manner, but to group them (to group size N_g) and then let all members in the group transform in parallel. The danger is now that neighbouring actions (each creating a valid part of the structure when executed alone), may together result in an invalid representation, that is, intersecting planes. For this the topology structure can be exploited when creating the parallel groups. The normal tGAP creation approach is followed: select the least important object, process this object, then select the next least important object, process this object and so on until enough objects in the group are found. However, when an object and the relevant direct neighbours for the generalization operation are selected, these will be temporarily marked together with their neighbours. If an object to be removed or one of its neighbours is already in the set of marked objects, then skip this object and continue with the next least important object (and in next grouping the skipped object will be first in line). By finding non-neighbouring objects, the local smooth-zoom actions (faces) will for sure not interfere.

Below a list of related research questions is presented:

- What is a good group size N_g? Too small approaches the normal tGAP, while too big N_g might result in a number of discrete stages (end of many non-vertical walls) at the same time. Also a too big N_g increases the chance on conflicts between neighbour actions. Having a larger group size N_g might also give a strange artificial effect during the smooth zoom: if many disappearing features are in the image, then some kind of artificial climax moment is introduced, when they all disappear together. Tests should be conducted whether this is noticeable (because only very few actions will be visible at the same time in one window).
- How often do groupings occur with real-world data for which the optimal group size N_g cannot be reached?
- Should the size of N_g vary due to: (1) stage in process (more to the top, smaller N_g; always be a percentage of the total number of objects at a given scale stage) or (2) relative to accumulated area change by actions in group?

4.2. Smooth zoom with real data

The first generalization operation introduced in the smooth tGAP was line simplification. One could wonder whether it is always possible to create a smooth tGAP structure and whether replacing the horizontal and vertical faces by a set of tilted faces will create errors (intersection faces). In Meijers (2011) it was proven that it is possible to perform a topology error-free simplification of the lines: starting with the guarantee that if there are no errors in the partition of the start scale, then there are also no errors in the resulting scale. Given this, it is then also possible to create a smooth tGAP (3D SSC) without any topology error which is also error-free at the intermediate scales. This is because the new, tilted faces, are always moving within the ‘free space’ and cannot intersect with other geometries.

Next question is whether it is always possible to realize a smooth merge without topology errors. The example data set, as used in Section 2, only shows very simple (convex) shapes to be removed and merged with its neighbour. It is easy to imagine that when there are two neighbouring equal rectangles, how one rectangle gradually has to take the space of the other rectangle and that the resulting non-horizontal faces in the smooth tGAP structure will be flat. The question that arises: is this always possible for any pair of arbitrarily shaped neighbours or configurations with island polygons included? Answer: yes, for strictly convex parts[1] of the disappearing area this is relatively easy, see Figure 11.

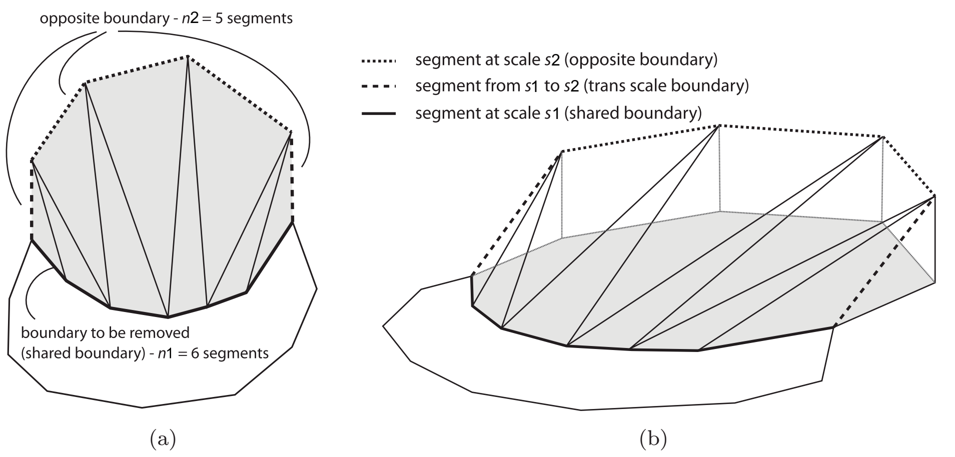
opposite boundary - $n2 = 5$ segments
segment at scale $s2$ (opposite boundary)
segment from $s1$ to $s2$ (trans scale boundary)
segment at scale $s1$ (shared boundary)
boundary to be removed
(shared boundary) - $n1 = 6$ segments
(a)
(b)

Figure 11. A way of constructing non-horizontal faces. In this example six triangles have their base in the shared boundary and $5 - 2 = 3$ triangles have their base in the opposite boundary. (a) Top view, (b) 3D space-scale view.

Figure 11 illustrates an approach to construct the faces between the boundary to be removed (shared boundary) and the boundary to be moved to (opposite boundary). This approach that requires convex parts is as follows: count the number of segments in the shared (n_1) and the opposite boundary (n_2). Now there will be n_1 triangles constructed, which will have their base in the shared boundary (and the remaining vertex on the opposite boundary) and $n_2 - 2$ triangles which will have their base in the opposite boundary (and the remaining vertex on the shared boundary). The triangles are created by iteratively taking simple step decisions (which triangle edge to add) in a duo-scan from start to end node over both boundaries: the shared and the opposite boundary. At every step there are two options: increment on shared boundary or increment on opposite boundary. The decision at every step is to take the step that corresponds to the shortest new edge added.

The 3D polyhedron corresponding to the vario-scale representation of a 2D area has different types of boundaries: at bottom (largest) and top (smallest) scale, these 2D boundaries of the 3D polyhedron are the two *fixed-scale* representations of this area. Between these scales, there are other 2D boundaries (in 3D space) connecting these fixed-scale representations. These boundaries are called the *trans-scale* boundaries. The combination of the fixed-scale and the trans-scale boundaries will result in a completely closed polyhedron.

Because of the convex shape, there will never be intersecting edges or faces. If the shape to be merged is concave, then decompose it in convex parts and treat the convex parts one by one. The order in which this should be done is to start with a direct neighbour part of the growing area (and repeat until all parts are processed); see Figure 12a. Note that this algorithm also works when the to be merged neighbour has an island: creating the strictly convex parts and processing these with the algorithm above will give correct results in the SSC; see Figure 12c.

Furthermore, this approach will also work as post-processing of a collapse/split operation: the split operation delivers boundaries to move to. Each part of a split polygon can, following the suggested approach, result in a gradual merge with its neighbour.

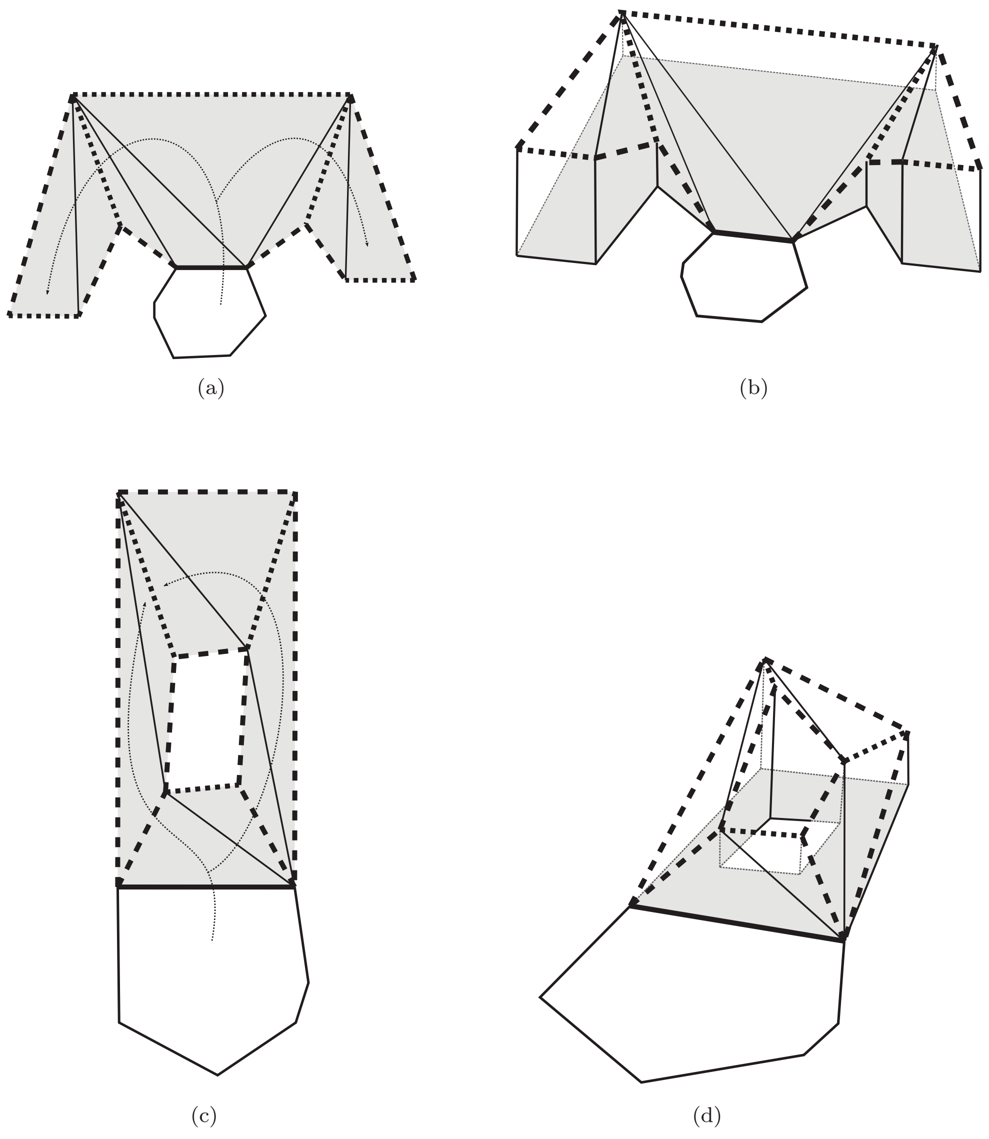

Figure 12. The processing of complex shapes into vario-scale representations: (a) the processing of a m-shaped neighbour, with growing area attached to middle leg of ‘M’ – top view, (b) 3D space-scale view, (c) the example of neighbour with island: decompose in strictly convex parts – top view, (d) 3D space-scale view.

5. The 3D smooth tGAP structure

Until now a base map was used. It is also possible to start with a base map (model) and then create in a similar manner a space-scale hypercube. The creation and use of the smooth tGAP structure is much the same as in 2D. In Subsection 5.1 it will be illustrated how the 4D hypercube can be created and also how a hyperplane in 4D space can be used to slice and result in a representation of 3D objects at the required homogenous scale (LoD). In Subsection 5.2, it will be explored how a 3D mixed-scale representation can be obtained by using non-horizontal slicing hyperplanes, i.e. slicing hyperplanes for which the scale value is not constant, but a function of the position.

5.1. Creation of 4D scale-space hypercube

To illustrate that the smooth tGAP is equally applicable to higher dimensional objects, we show the following 3D example leading to a 4D hypercube.

Figure 13a and d, respectively, show a higher and a lower-detailed 3D object representation. In the higher-detailed 3D representation objects *zi* and *zii* are each represented as 3D-objects (n-dimensional). The objects *zi*, *zii* are delimited by 2D boundaries (($n-1$)-dimensional) having at least one boundary segment. In this case each object is delimited by its set of faces, front face, back face, left-side face, right-side face, bottom face and top face. Note that object *zii* has seven faces, because the left face has two parts: one of which is shared with object *zi*. These faces form the boundary segments.

Figure 13a shows a set of two 3D objects having the highest LoD. Object *zi* represents, for example, a first building, object *zii* represents a second building. Also a lower LoD 3D object representation is generated as shown in Figure 13d. In this lower-detailed 3D object representation the original objects *zi* and *zii* are merged into object *ZII*. Figure 13b and c show subsequent intermediate 3D representations. One could argue that this is a simple case, and the question remains if such a smooth transformation is also possible in an arbitrary 3D situation of merging a disappearing 3D object into its neighbour. Therefore, Figure 14a–c show how a topological correct mapping (gradual transition) is achieved using a generic approach as further explained below. A generic gradual transition is shown with a growing object *zii* and a shrinking and gradually disappearing object *zi*.

Figure 14a again shows the higher-detailed 3D object-presentation as in Figure 13a. Objects *zi*, *zii* are convex objects[2] that have a shared boundary formed by the right-side face (dark) of object *zi*. Object *zi* is to be removed in a merge operation with growing object *zii*. In the lower-detailed 3D object representation of Figure 13d, this shared boundary is mapped to a destination boundary comprising bottom face one, top face three, front face two, back face four and left-side face five as indicated in Figure 14a. As shown in Figure 14b, the shared boundary between the objects *zi* and *zii* is now partitioned into boundary segments to equalize the number of faces, edges and nodes in the shared boundary and the destination (or opposite) boundary. In the example shown the shared boundary is provided with additional nodes a, b, c, d that are mapped to nodes A, B, C, D, edges $a-b$, $b-c$, etc., mapped to $A-B$, $B-C$, etc., and faces one to five that are mapped to faces one to five of the destination (opposite) boundary. Lowercase and uppercase characters indicate elements in the higher-detailed 3D representation and in the lower-detailed 3D-representation, respectively.

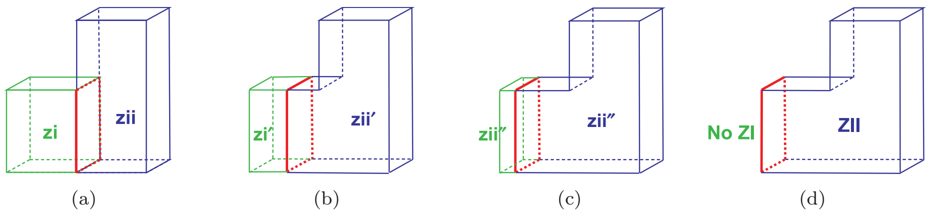

Figure 13. A simple 3D scene: the more important object *zii* gradually takes over the less important object *zi*. (a) initial configuration, (b) at 1/3rd of the gradual transition, (c) at 2/3rd of the gradual transition, (d) final configuration.

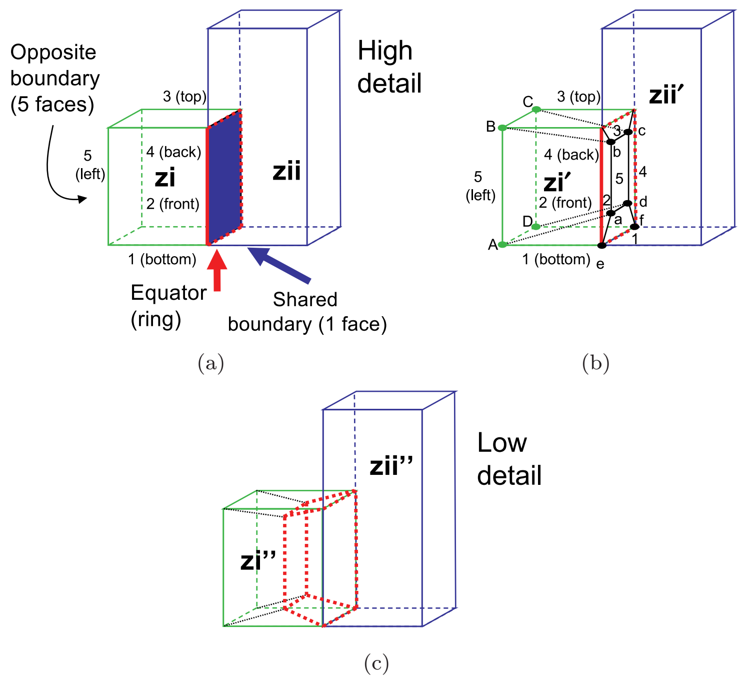

Figure 14. Alternative manner for gradually taking over the space of the less important object by the more important object. In this case the trans-scale boundaries stay as long as possible inside the less important object *zi''*. (a) initial configuration, (b) mapping of opposite boundary to share boundary, (c) halfway the gradual transition.

A 4D object representation is constructed, that has in addition to the spatial dimensions of the 3D object representations an additional scale dimension integrated. The higher and the lower-detailed 3D object representations are assigned a first and a second value ($s1$, $s2$ of scale level s) for the scale dimension, respectively. For example, nodes a, b, c, d of the object *zii* in the higher-detailed representation are defined by their 3D spatial coordinates x, y, z and a value $s1$ for the scale coordinate s. For example, nodes A, B, C, D in the lower-detailed representation are defined by their 3D spatial coordinates x, y, z and a value $s2$ for fourth coordinate s. These two 3D representations at fixed scales $s1$ and $s2$ form the first two boundaries of the 4D representation (the 'fixed scale' boundaries).

Next a trans-scale boundary is constructed that is delimited between mutually corresponding 2D boundary segments of one object in the higher-detailed and the lower-detailed 3D representation. In this case the 2D boundary segments of object *zii* (faces) in the higher-detailed representation are the top, bottom front, back, right and two left-side face of object *zii*. The lower left-side face of object *zii* forms a shared boundary with object *zi* and a remaining portion, not shared with object *zi*. The shared portion is partitioned in faces one to five that are mapped to faces one to five of the destination (opposite) boundary as indicated above. Each pair of a face one to five and its corresponding face one to five to which it is mapped delimits a trans-scale n-dimensional boundary segment in $(n + 1)$-dimensional space. For example, consider face one in the shared boundary, which is defined by nodes a, d, f, e in the higher-detailed representation. This face one is mapped to corresponding face one in the destination boundary defined by A, D, F, E in the lower-detailed representation. For clarity labels E and F are not shown in the drawing. Node E has the same values for the coordinates x, y, z as its corresponding node e, but only has a different scale value, $s2$ instead of $s1$. Likewise node F only differs by its scale value from node f. The trans-scale boundary segment delimited by face one: a, d, e, f in the higher-detailed representation and its corresponding face one: A, D, E, F in the lower-detailed representation comprises four other faces: a first face a, d, D, A; a second face d, f, F, D; a third face f, e, E, F and a fourth face e, a, A, E. Together these faces form one of the 3D boundaries (n-dimensional) of the 4D representation ($(n+1)$-dimensional). Analogously, a trans-scale boundary segment is constructed that is delimited between each of the other faces two to five in the higher-detailed object representation and its corresponding one of the other faces in the lower-detailed object representation. Further, analogously respective trans-scale boundary segments are constructed that each are delimited between each of the other faces of object *zii* not part of the shared boundary in the higher-detailed object representation and their corresponding one of the other faces in the lower-detailed object representation.

The concatenation of all trans-scale ($n = 3$)-dimensional boundary segments with the two 3D fixed scale (boundary) representations of the object (*zii* at scale $s1$ and *ZII* at scale $s2$) forms the vario-scale ($n + 1 = 4$)-dimensional representation associated with the object *zii*–*ZII*.

An intermediate 3D representation for object *zii* is now obtained by calculating a cross-section between a 3D slicing hyperplane (object) and the constructed 4D representation, wherein the cross-section of said trans-scale boundary with the slicing object forms the boundary of the corresponding object assigned to said object in the intermediate 3D representation.

Analogous to the 2D case the slicing object may be formed by a horizontal hyperplane (3D hyperplane in 4D space), that is, an object according to the definition s = constant, wherein said constant value is a value in the range between $s1$ and $s2$. Alternatively the value s may be a function of one or more of the coordinates x, y, z, so that the intermediate representation has a LoD that is position dependent; e.g. to support the generation of 3D perspective views (non-horizontal slicing hyperplanes); see Subsection 5.2.

Figure 14c shows an example of an intermediate representation obtained, in the generic general transition, when the slicing object has a value s equal to $\frac{s1+s2}{2}$. In the so obtained intermediary representation the original boundary is extended with the volume indicated by dotted lines. One could argue that the general approach results in less attractive intermediate representations, compared to the simple approach of Figure 13b and c. However, the point of the generic approach is to proof that it is always possible to have a gradual transition. Another advantage of the generic approach is that it does not affect the faces, edges and nodes on the non-shared outside surface patches of the disappearing object. So, no topology problems will occur with third (non-depicted) objects

For clarity the present example is on purpose set out for a simple case, where only a limited number of objects is involved and it is shown how the trans-scale boundary is calculated for a single object. In practice the higher and the lower-detailed n-dimensional representation may represent plurality of objects; e.g. forming a partition of space (at every scale). The step of determining the trans-scale boundary and the step of calculating the cross-section is executed for each of the objects. In practice a boundary segment of an object is shared with another object. Accordingly once a trans-scale boundary element delimited by the boundary segment in the higher-detailed representation and its corresponding boundary segment in the lower-detailed representation is calculated for an object, it can be reused for the other object sharing said boundary element.

Figure 15a –c show examples of a simplify operation in the 3D case. Figure 15a shows a merged object *ZII* obtained by a merging operation as illustrated with reference to

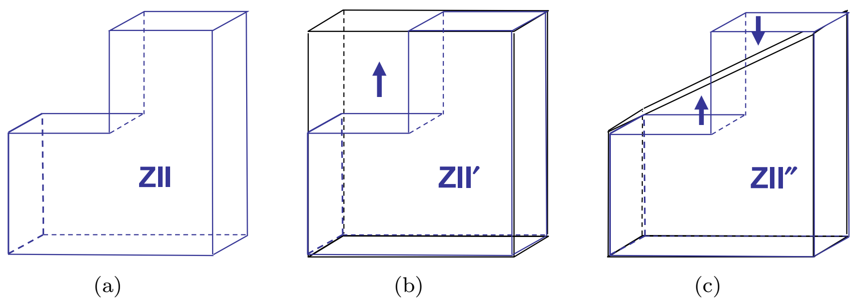

Figure 15. Two alternative options for the boundary simplification of the object ZII: (a) two merged objects (*ZII*), (b) simplification, keep block shape, (c) alternative simplification, tilted roof shape.

Figure 13a–d and with reference to Figure 14a–c. Figure 15b shows a result according to a simplify operation of a first type, wherein the object is approximated by its 'bounding box' (bbox), i.e. the smallest cuboid that contains the object. Figure 15c shows a result according to a simplify operation of a second type, wherein the object is otherwise approximated (with a tilted roof). Note that the non-depicted neighbour volume objects are affected.

16a and b show a pseudo 4D impression for the merge followed by one of the two simplification options. In this view it is shown how an $(n+1)$-dimensional object representation is constructed, having in addition to the geometric dimensions x, y, z, the additional scale dimension s. Representations are assigned a first and a second value $s1$ and $s2$ for the additional scale dimension, respectively. The higher-detailed 3D representation for object *zii* is assigned scale value $s1$ in the 4D object representation and the lower-detailed 3D representation for the resulting merged and simplified object *ZII′* (first type of simplification) is assigned scale value $s2$ in the 4D object representation. The dotted lines indicate the relation between mutually corresponding nodes in the representations for $s1$ and $s2$. Figure 16b is the same as Figure 16a only with a different simplification operation applied, leading to *ZII″* instead of *ZII′*.

5.2. Using the 4D hypercube to derive 3D mixed-scale representations

The result of the process described in the previous subsection is a 4D data structure, based on a partition of the integrated 3D space and scale representation (4D SSC). It should be noted that this is not only a representation in a higher dimensional (4D) space, but that also the higher dimensional primitives are really used. A 4D primitive in this structure corresponds to a vario-scale representation of a (3D) real-world object. The boundaries of this 4D primitive are formed by two 3D representations at start and end scale (higher and lower-detailed representations), which are connected by trans-scale boundaries (3D primitives).

Note that perhaps the mixed-scale representation might be a bit uncommon for 2D data, but this could be applied a lot in the case of 3D data. A perspective view in 3D computer graphics (for CAD systems or 3D games) would be very well served with such a mixed-scale representation. However, all known 3D computer graphics methods are based on a fixed number of LoDs. The drawback is that there is not only redundancy in these LoDs, but that when going from one LoD to the next LoD there are always 'fitting' problems. In fact these are topology errors in the 'mixed' scale representation based on

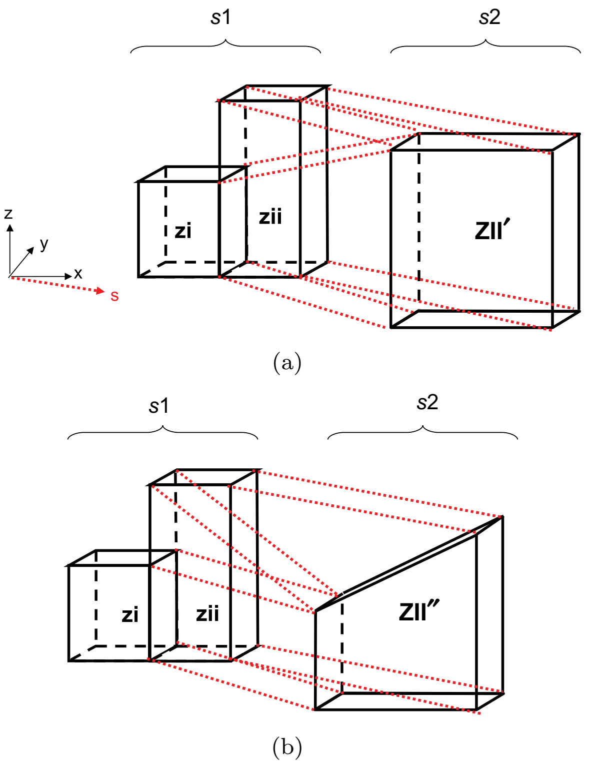

Figure 16. A pseudo 4D illustration of our simple 3D scene. Object *zi* is only shown for reference purposes. The red lines illustrate the vario-scale representation of the second object; *zii* is gradually changed into *ZII′* and *ZII″* (depending on which simplification option is chosen – bbox or tilted roof). (a) bbox simplification, (b) tilted roof simplification.

multiple LODs. These sliver errors, small overlaps or gaps in the transition between two LoDs, distract the user (viewer) of the 3D data during the interaction. The smooth tGAP structure (4D) enables to obtain a 3D mixed-scale representation without topology errors: a perfect partition of the 3D space. This is achieved by slicing with a tilted 4D hyperplane.

The 4D space-scale hypercube can be used for good perspective view visualizations by taking non-horizontal scale slices: near to the viewer a lot of detail (large scale) far away from the viewer not so much detail (small scale). The intersection of this 4D hypercube with the 3D hyperplane gives a perfect 3D topology: all representations do fit without gaps or overlaps. This solves a big problem as often the case in the transition from one LoD to the next LoD in computer graphics. Interesting ‘implementation’ issues will arise: How can the slicing in the 4D hypercube be done efficiently? Is this efficient enough for interactive rendering performance (50–100 times per second)? The result of a slicing operation is a 3D model and still has to be rendered on a 2D display (or 3D stereo device). Note that we should attempt to apply hyperplane-sweep methods (to avoid unneeded computations). Would it be possible to combine the above two steps in a single operation on the 4D hypercube (selection and transformation for display)? What steps can be done in hardware and what needs to be done in software?

6. Conclusion and future work

In this section first the main results of our research are presented and then the paper is concluded with a long list of ongoing and future work, aiming to resolve the open questions.

6.1. Main results

This paper has introduced the first true vario-scale structure for geographic information: a delta in scale leads to a delta in the map continuously down to an infinitesimal small delta for all scales. The smoothness is accomplished by removing all horizontal faces of the classic tGAP structure. Algorithms and corresponding intuitive proofs of correctness were given how to create the smooth tGAP structure performing generalization operations in such a way that the output given gradually changes the boundaries between the features being generalized.

The smooth tGAP structure delivers true vario-scale data and can be used for smooth zoom. The proposed new structure has very significant advantages over existing multi-scale/multi-representation solutions (in addition to being truly vario-scale): (1) due to tight integration of space and scale, there is guaranteed consistency between scales (it is one integrated space-scale partition) and when using non-horizontal slices the resulting 2D maps will be a valid, mixed-scale planar partition: this is useful in 3D computer graphics, (2) it is relatively easy to implement smooth zoom, and (3) it provides a compact (good for storage, transfer and CPU use) and object-oriented encoding (one higher dimensional object for a complete scale range). Furthermore, we illustrated that the smooth tGAP is equally applicable to higher dimensional objects (i.e. integration of 3D space and scale leading to a 4D hypercube).

6.2. Open research questions

Although the smooth tGAP structure is a breaktrough vario-scale data structure supporting smooth zoom, there is still a myriad of open research questions:

- Engineering: Determine how to encode the space-scale (hyper) cube in an efficient manner? This entails selecting the proper technical tools (programming languages and libraries) for producing a prototype. Also creating metrics and collecting statistics (comparing them to normal 2D storage of the largest scale map (at a single scale), either with or without a topological structure), for example, how many nodes, edges, faces and volumes in the SSC (and which primitives and references explicitly stored, van Oosterom *et al.* 2002, Meijers *et al.* 2009)? An important decision is whether to encode the SSC with or without a topology structure. As the SSC is a full partition of the SSC, it seems attractive to use a topology encoding because this allows efficient navigation through the data, avoiding redundancy (double storage), guaranteeing consistent data (no gaps or overlaps) and fitting very well to the tGAP structure.
- Formalize the structure and prove that all claims can indeed be backed by sound mathematical proofs instead of the more intuitive proofs as presented in the current paper. To be based on Meijers and van Oosterom (2011) and Thompson and van Oosterom (2012).
- Tune the process: implementation of the smooth tGAP structure takes two main steps: (1) build classic tGAP and (2) transform from classic to smooth tGAP (SSC).

The smooth tGAP has the same building challenge as the classic tGAP with respect to applying the right sequence of generalization operators (remove or merge, collapse or split, simplify) to obtain cartographic quality. This has to be well tuned, otherwise the maps will be of (too) low cartographic quality despite the fact that they are perfect in topological sense and 100% consistent between scales. One option for this might be the constrained tGAP (Haunert *et al.* 2009). It is also clear that this requires 'understanding' (semantics) of the different types of object classes involved (and the mapping needs of the end-users).

- Test with larger real-world data sets and appropriate graphical user interfaces supporting smooth-zoom visualization, mixed-scale visualization and observing end-user behaviour (this is a typical Human Computer Interaction study). The vario-scale structure and the smooth-zooming principles imply that in a transition some of the cartographic principles of legibility are temporarily set aside (minimum size and minimum distance). Probably different devices/platforms (desktop, mobile) have to be tested and users have to be given a range of relevant tasks. Such research should find out what is the optimal height for a given device and scale to slice the SSC. Large data sets result in large cubes, a slice near the bottom will contain a lot of data which takes time and is not what a user wants. So slicing as explained in Subsection 3.2 should be combined with other (spatial) selection criteria; e.g. the bbox. The bbox is most likely smaller at the bottom and larger near the top for 'sane' applications. For non-horizontal slices the lower edges of the bbox should be shorter than the higher edges of the bbox. This can be compared to the use of frustums in 3D computer graphics for perspective views.
- Further investigate the effects of the collapse/split operator in the smooth tGAP structure. After collapse an area object lives on as a line (or point) object. In the SSC this object is then represented by a polyhedral volume to which a vertical surface is connected at the top (in case of a collapse to a point then in the SSC the polyhedral object is extended with a vertical line). Note that these objects have a non-manifold boundary. All attributes are attached to the same object, which is represented in the SSC by connected multiple parts of, respectively, dimension three and two in case of a collapse to a line and one in case of a collapse to a point.
- Improve the cartographic generalization quality of the smooth tGAP structure. This is an important ongoing goal, and to reach this goal, not only more refined use based on the object classification and better application of the currently implemented operators (delete, merge, collapse, split), but also the implementation of more generalization operators in the smooth tGAP structure will be explored. The operators of which we think that it would be possible to implement them within the smooth tGAP structure include displacement, typification and symbolization (Cecconi 2003).

Furthermore, we would like to explore the use of the smooth tGAP structure for different usage contexts (for different types of users and their needs). The smooth tGAP could be compared to an index in a relational database: it is an additional structure (to the base table) enabling efficient access according to some kind of selection criteria. For example, on a single table with person data it is possible to define an index on name, another index on date of birth and yet another index on location. Same would be true for a single set of base data: one smooth tGAP structure could be created on top of this base data set for users interested in urban planning, while another smooth tGAP structure could be created for users interested in navigation of the waterways.

- Despite the fact that the proposed solution results in a true vario-scale structure, it has still an (old) tGAP drawback and that is the one by one sequencing of all generalization operations. This might give a suboptimal smooth-zoom effect – more experiments with end-users are needed to verify whether this is indeed suboptimal. A solution for the one by one sequencing is not to implement the steps in the structure in a sequential manner, but to group them and let all members in the group transform in parallel. In Subsection 4.1 some possible strategies to realize more parallel actions in the tGAP creation were proposed. These need further exploration.
- Test map-overlay processing with two (or more) independent SSCs – this 3D overlay resembles conventional data integration in that it is possible to geometrically overlay the two SSCs and carry over the attribute information to the newly segmented space-scale partition. Note that before the actual overlay, the scale dimension first has to be well aligned so that only the corresponding representations intersect. However, for data integration this will not be enough, one of the difficulties will be to harmonize semantically the attribute values. Using SSCs might give more clues for a data integration process than just integrating two separate 2D map sheets (e.g. which do not have the same reference scale) and can be helpful for performing both horizontal as well as vertical conflation at the same time. An example application could be creating a smooth tGAP based on a soil map 1:50,000 and a land cover map 1:100,000. Intersect the two SSC and use the result to answer the request to find the areas that are forest on sandy soils at scale 1:250,000.
- The smooth tGAP structure is mainly a DLM (digital landscape model) based representation, which is supporting application of cartographic styling at the last moment before display (Stoter *et al.* 2010). Due to the specific nature of the smooth tGAP, the cartographic styling requires additional attention. For example, when a road is an area on the largest scale and in the tGAP structure the road area is collapsed to a road centerline via completely smooth transition, then in the visualization a shock might appear if the road area is displayed by colouring the area (which becomes infinitely thin before it is represented by a line. The line is also infinitely thin, but displayed with line-symbology, to make it visible. One possible approach to avoid this to display 'shock', is not simply colouring the road area, but also provide the proper casing of this area, which might even be of the same colour.
- Another important aspect of maps is including labels in the presentation. In a similar manner as a 'delta scale, delta map' rule applies to the geographic features, the same applies to the labels. During zooming (and panning) the users should not be confronted with shocks in the label display. In Been *et al.* (2010) this is called the 'Dynamic Map Labeling' problem and the paper also applies the paradigm that scale is considered as additional dimension (as in our smooth tGAP represented by the SSC). We want to further explore how their smooth label approach fits in our SSC. Note that this might mix DLM and DCM (digital cartographic model) related concepts inside the structure.
- In this paper it was assumed that the most detailed representation is defined by linear primitives: boundaries in 2D maps are straight line segments (and in 3D models planar faces). However, in existing large scale maps also non-linear primitives are used; in the Netherlands circular arcs are used a lot in 2D large scale topographic maps. Similarly, one might expect that in 3D models also non-linear surfaces are used to represent the boundaries of 3D volume objects; cylinder or

sphere patches. One could also imagine the use of non-uniform rational B-splines (NURBS) to represent the shape of curved surfaces (Pu and Zlatanova 2006); e.g. a civil engineer designs a dike or embankment (in Dutch: 'talud') with NURBS. In theory also these non-linear primitives can be used in the smooth tGAP structure. However, how to apply these in practice, both when creating and using the structure, results in several engineering challenges.

- Make the structure dynamic: currently the tGAP structure (including the new smooth tGAP) is a static structure. When an update takes place, the structure has to be recomputed. Due to global optimization criteria, the impact of a local change is not guaranteed to have a local effect; e.g. limited to path in structure from changed object to root of structure, perhaps including sibling. The grouping approach of Subsection 4.1 might be helpful for making more local updates possible. Making the structure dynamic can result in a 5D hypercube (van Oosterom and Stoter 2010) with an additional temporal dimension. Again slicing issues arise when we want to create visualizations: slice from 5D to 4D with hyperplane (e.g. select a specific moment in time or alternatively select a specific scale).

Acknowledgements

This research is supported by the Dutch Technology Foundation STW (project numbers 11,300 and 11,185), which is part of the Netherlands Organisation for Scientific Research (NWO) and partly funded by the Ministry of Economic Affairs.

We would like to thank the anonymous reviewers for their concise reading and very constructive comments, which have improved the paper. Also thanks to Rod Thompson for proofreading the paper and suggesting numerous improvements. The authors would further like to thank Dirk de Jong, European Patent Attorney at Vereenigde, for the inspiring questions and discussions during the process of writing the patent claim for the method and system description of true vario-scale maps (patent pending nr. OCNL 2006630). All (remaining) errors are the sole fault of the authors.

Notes

1. A simple polygon is strictly convex if every internal angle is strictly less than 180° (so not equal to 180°).
2. In case of a concave object, this is then first decomposed into its convex parts, similar to the 2D approach.

References

Baars, M., *et al.*, 2004. Rule-based or explicit storage of topology structure: a comparison case study. *In*: F. Toppen and P. Prastacos, eds. *Proceedings of the 7th conference on geographic information science [CD-ROM]*. Heraklion: Crete University Press, 765–769.

Been, K., *et al.*, 2010. Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry*, 43, 312–328. Special Issue on 24th Annual Symposium on Computational Geometry (SoCG' 08).

Cecconi, A., 2003. Integration of cartographic generalization and multi-scale databases for enhanced web mapping. Thesis (PhD). University of Zurich.

Cecconi, A. and Galanda, M., 2002. Adaptive zooming in web cartography. *Computer Graphics Forum*, 21, 787–799.

Hägerstrand, T., 1970. What about people in regional science?. *Papers in Regional Science*, 24, 6–21.

Hampe, M., Sester, M., and Harrie, L., 2004. Multiple representation databases to support visualization on mobile devices. *In*: O. Altan, ed. *Proceedings of the XXth ISPRS Congress*, *XXXV of*

International archives of photogrammetry, remote sensing and spatial information sciences, July, Istanbul, Turkey. Enschede: ISPRS, 135–140.

Harrie, L., Sarjakoski, L.T., and Lehto, L., 2002. A variable-scale map for small-display cartography. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34, 237–242.

Haunert, J.H., Dilo, A., and van Oosterom, P., 2009. Constrained set-up of the tGAP structure for progressive vector data transfer. *Computers & Geosciences*, 35, 2191–2203. Progressive Transmission of Spatial Datasets in the Web Environment.

Meijers, M., 2011. Simultaneous & topologically-safe line simplification for a variable-scale planar partition. *In*: S. Geertman, W. Reinhardt, and F. Toppen, eds. *Advancing geoinformation science for a changing world*. Lecture notes in geoinformation and cartography. Berlin: Springer, 337–358.

Meijers, M. and van Oosterom, P., 2011. The space-scale cube: an integrated model for 2D polygonal areas and scale. *In*: E.M. Fendel, *et al.,* eds. *Proceedings of the 28th urban data management symposium*, *Vol. 38 of International archives of photogrammetry*, *remote sensing and spatial information sciences*, September. Enschede: ISPRS, 95–102.

Meijers, M., van Oosterom, P., and Quak, W., 2009. A storage and transfer efficient data structure for variable scale vector data. *In*: M. Sester, L. Bernard, and V. Paelke, eds. *Proceedings of the advances in GIScience*, Lecture Notes in Geoinformation and Cartography. Berlin: Springer, 345–367.

Noguera, J.M., *et al.*, 2010. Navigating large terrains using commodity mobile devices. *Computers & Geosciences*, 37, 1218–1233.

Nöllenburg, M., *et al.*, 2008. Morphing polylines: a step towards continuous generalization. *Computers, Environment and Urban Systems*, 32, 248–260. Geographical Information Science Research – United Kingdom.

Penninga, F., 2004. Oracle 10g Topology; Testing Oracle 10g Topology using cadastral data. Technical report, Delft University of Technology, Delft.

Pu, S. and Zlatanova, S., 2006. Integration of GIS and CAD at DBMS level. *In*: E. Fendel and M. Rumor, eds. *Proceedings of UDMS' 06 Aalborg*. Delft: Urban Data Management Society, 9.61–9.71.

Sester, M. and Brenner, C., 2005. Continuous generalization for visualization on small mobile devices. *In*: P. Fisher, ed. *Developments in spatial data handling*. Berlin: Springer, 355–368.

Stoter, J., *et al.*, 2010. Applying DLM and DCM concepts in a multi-scale data environment. *In*: B. Buttenfield, *et al.*, eds. *Proceedings of GDI 2010: symposium on generalization and data integration*. Boulder, CO: USGS-CEGIS/University of Colorado/Penn State University/University of California, 1–7.

Thompson, R.M., and van Oosterom, P., 2012. Modelling and validation of 3D cadastral objects. *In*: S. Zlatanova, *et al.*, eds. *Proceedings of the urban and regional data management – UDMS annual 2011*, September. Leiden: CRC Press, 7–23.

van Kreveld, M., 2001. Smooth generalization for continuous zooming. *In*: D. Du, *et al.*, eds. *Proceedings of the 20th International Cartographic Conference* (*ICC '01*), 6–10 August 2001, Beijing, China, 2180–2185.

van Oosterom, P., *et al.*, 2002. The balance between geometry and topology. *In*: D. Richardson and P. van Oosterom, eds. *Proceedings of the advances in spatial data handling*, *10th international symposium on spatial data handling*, 8–12 July 2002, Ottawa, Canada. Berlin: Springer, 121–135.

van Oosterom, P., 2005. Variable-scale topological data structures suitable for progressive data transfer: the GAP-face tree and GAP-edge forest. *Cartography and Geographic Information Science*, 32, 331–346.

van Oosterom, P. and Stoter, J., 2010. 5D data modelling: full integration of 2D/3D space, time and scale dimensions. *In*: S.I. Fabrikant, *et al.*, eds. *Proceedings of the 6th international conference on geographic information science*, *GIScience' 10*, Zurich, Switzerland. Berlin: Springer, 310–324.

Vermeij, M., *et al.*, 2003. Storing and using scale-less topological data efficiently in a client-server DBMS environment. *In*: D. Martin, ed. *Proceedings of the GeoComputation 2003*, 8–10 September 2003, University of Southampton, Southampton.