VARIO-SCALE DATA STRUCTURES FOR 2D AND 3D GEO-INFORMATION[1]

[1]Peter van Oosterom, [1]Martijn Meijers

[1]Delft University of Technology, OTB Research Institute for the Built Environment
P.O. Box 5030, 2600 GA Delft, THE NETHERLANDS.

email: [1]P.J.M.vanOosterom@tudelft.nl and [2]B.M.Meijers@tudelft.nl

Abstract

This paper introduces the concept of the smooth tGAP structure represented by a space-scale partition, which we term the space-scale cube. We take the view of 'map generalization is extrusion of data into an additional dimension'. For 2D objects the resulting vario-scale representation is a 3D structure, while for 3D objects the resulting vario-scale representation is a 4D structure. This paper provides insights in: 1. creating valid data for the cube and proof that this always possible for the implemented 2D tGAP (topological Generalized Area Partition) 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.

Keywords: *vario-scale, generalization, progressive transfer, smooth zoom*

1 INTRODUCTION

Technological advancements have lead to maps being used virtually everywhere; e.g. 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. Although there are some useful efforts (van Kreveld, 2001; 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. From the structure continuous generalizations of real world features can be derived and can be used for presenting a smooth zoom action to the user. Furthermore, mixed-scale visualizations can be derived (more and less generalized features shown together in one visualization), in which the objects are consistent with each other. Making such a transition area is mostly one of the difficulties for 3D computer graphic solutions (e.g. using stitch strips based on triangles, like in Noguera et al., 2010).
The remainder of this paper is structured as follows: Section 2 contains a discussion how the classic tGAP structure can be represented by a 3D space-scale cube and how this can adapted

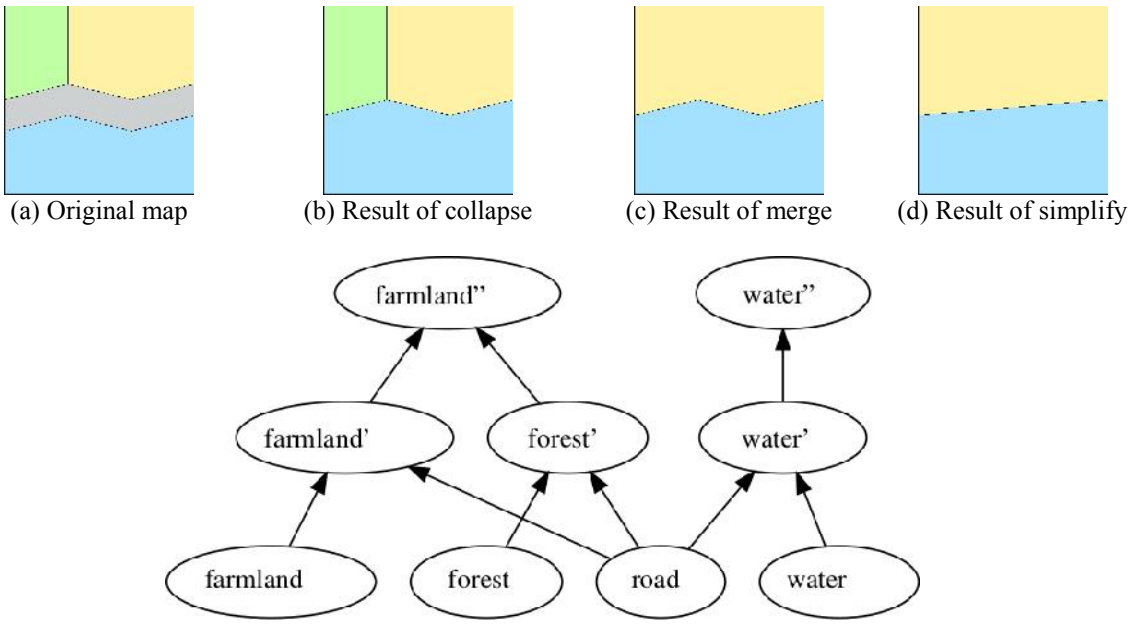
[1] This paper is also published in the annual report of the Royal Netherlands Academy of Arts and Sciences/ Netherlands Geodetic Commission (KNAW/NCG), Delft, 2012, pages 21-42. ISBN: 978 90 6132 339 6.

to store more continuous generalization. The use and application of the smooth tGAP structure is further discussed in Section 3. Section 4 proofs 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 space-scale cube 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 (and 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 to efficiently select a representation at any required level of detail (scale or importance). Figure 1 shows 4 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 the road and assign parts to neighbours);
2. Remove forest area and merge free space into neighbour farmland;
3. Simplify boundary between farmland and water area.



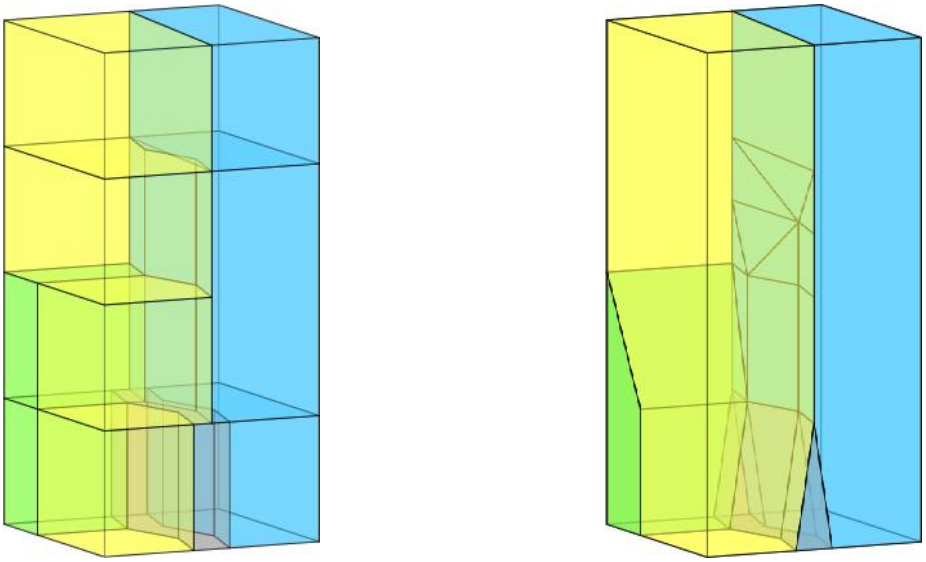
(a) Original map (b) Result of collapse (c) Result of merge (d) Result of simplify

(e) Corresponding tGAP structure

Figure 1: The 4 map fragments and corresponding tGAP structure

2.1 The tGAP structure represented by the 3D space-scale cube

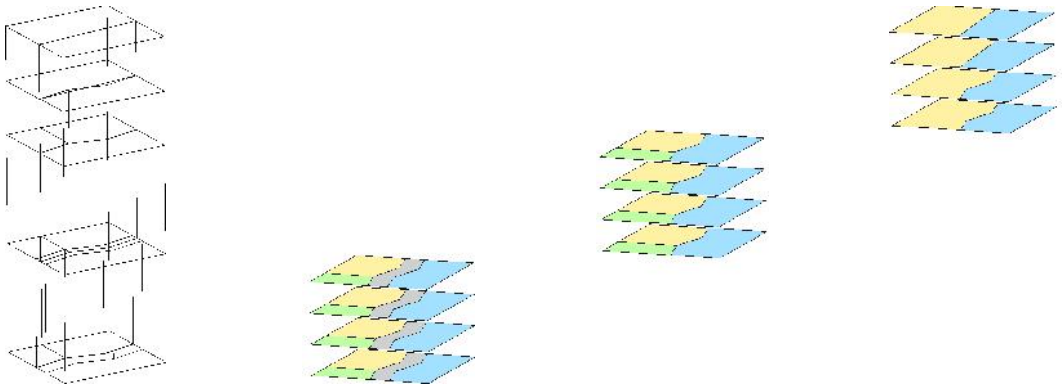
The tGAP structure is a DAG and not a tree structure, as the split causes the road object to have several parents; see Figure 1(e). In our current implementation the simplify operation on the relevant boundaries is combined with the remove or collapse/split operators and is not a separate step. However, 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 space-scale cube (SSC) representation (Vermeij et al., 2003; Meijers and van Oosterom, 2011). We termed this representation the space-scale cube in analogy with the space-time cube as first introduced by Hägerstrand (1970). Figure 2(a) shows this 3D representation for the example scene of Figure 1. In the SSC the vario-scale 2D area objects 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 to a line, the vario-scale representation consist of a compound geometry with a 3D volume-part and 2D face-part attached.



(a) SSC for the classic tGAP structure (b) SSC for the smooth tGAP structure

Figure 2: The space-scale cube (SSC) representation in 3D

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 do cause a sudden local 'shock': a small scale change results in a not so small geometry change; e.g. leading to complete objects disappearing; see Figure 3. In the space-scale cube this is represented by a horizontal face; a sudden end or start of 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 the collection of fitting prism shapes, a full partition of the space-scale cube).



(a) Wireframe of (classic) space-scale cube (b) Slices for Step 1 (c) Slices for Step 2 (d) Slices for Step 3

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

2.2 Smooth line simplification

In order to obtain more gradual changes when zooming, i.e. in a morphing style (c.f. Sester and Brenner, 2005; Nöllenburg et al., 2008), we first realised that the line simplification operation could also output non-vertical faces for the space-scale cube 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 space-scale cube; see Figure 4. Note that both the sudden-change line simplification and the gradual-change line simplification have both 3 faces in the SSC: sudden-change has 2 rectangles and 1 triangle and gradual-change has 3 triangles. When slicing a map based on the SSC fragment as depicted in Figure 4(b) (to 'slice' means taking a cross-section of the cube) at a certain scale, a delta in scale leads to a derived delta in the map. That is, a small change in the geometry of the depicted map objects and no sudden change any more, as was the case with the horizontal faces parallel with the bottom of the cube, which were the results of the merge or split operations. Note that 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.



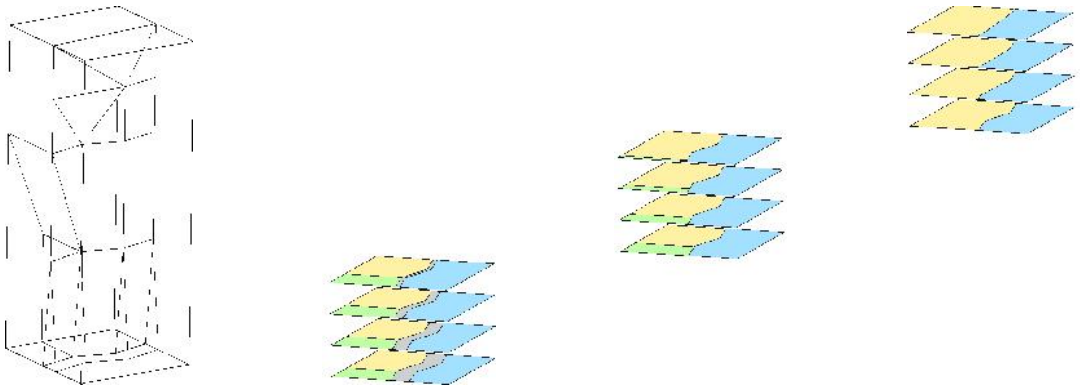
(a) Sudden-change line simplification: 2 rectangles and 1 triangle (b) Gradual-change line simplification:: 3 triangles

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

2.3 Smooth merge and split

The merge and split (collapse) operations can, similar to 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 space-scale cube this corresponds to non-vertical faces (and there is no more need for a

horizontal face, i.e. a suddenly disappearing feature); see Figure 2(b). 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, e.g. the big front-right face in Figure 2(b) corresponds to four faces in Figure 2(a). 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 is that the merging of the different polyhedral volumes belonging to the same real world object is that also the number of volumes is reduced: there is a one-to-one correspondence between a single object and its smooth tGAP polyhedral representation, valid for all relevant map scales. The benefit of a smaller number of primitives, the nodes, edges, faces and volumes, is that there are also less 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 Louwsma et al., 2003; Baars et al., 2004; Penninga, 2004). This is even more true for topology based vario-scale data structures (c. f. Meijers et al., 2009). Lighter structures are more suitable for (progressive) data transfer and high(er) performance. Figure 5 illustrates the resulting true vario-scale structure: small deltas in scale will give small deltas for map areas.



(a) Wireframe of (smooth) space-scale cube (b) Slices for Step 1 (c) Slices for Step 2 (d) Slices for Step 3

Figure 5: The map slices of the smooth tGAP structure: (b) step 1 (collapse), (c) step 2 (merge) and (d) step 3 (simplify). Note the continuous changes, also in between the 'true' tGAP events.

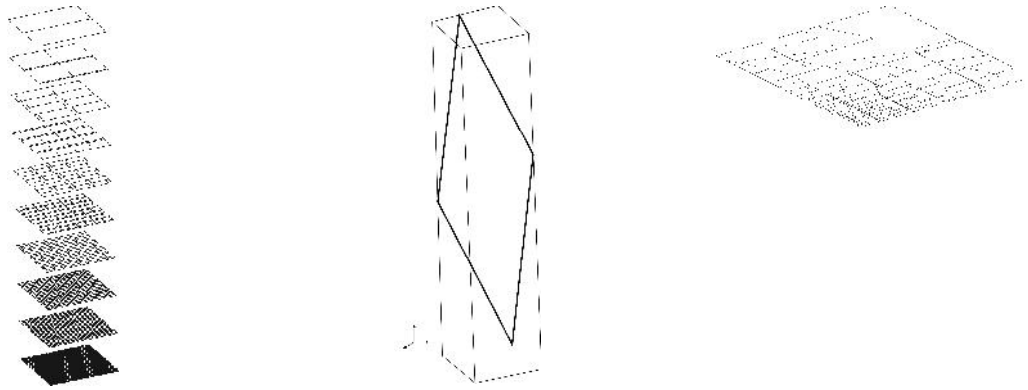
3 ADVANCED USE OF THE SMOOTH tGAP

In this section a number of more advanced usages of the smooth tGAP structure will be discussed. That is using the SSC for more than only for horizontal slices at an arbitrary scale to create a representation which is homogenous with respect to the map scale. First, it will be illustrated how a mixed scale representation can be obtained from the smooth tGAP structure in Subsection 3.1. The next subsection 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 parallel to the bottom and top of the cube were discussed and used for creating 2D maps with homogenous scale. It is not strictly necessary to do parallel slices, nothing prevents from taking non-horizontal slices. Figure 6 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 side where slice is close to the bottom of the cube, less detail at the side

where slice is closer to the top. Compare to 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 features far away (intended for display on small scale) and less generalized features close to the observer (large scale).



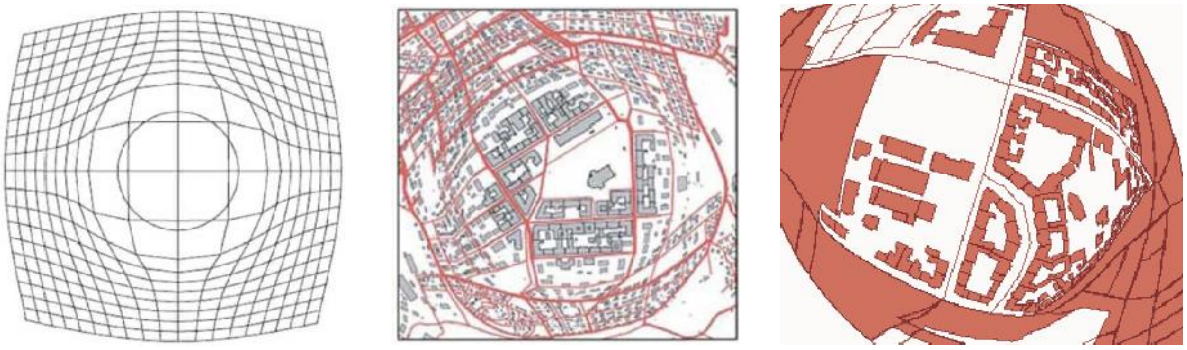
(a) A set of smooth slices derived from the SSC

(b) How the non-horizontal slice of (c) is taken.

(c) Corresponding mixed-scale map (non-horizontal slice): top of map shows more generalized features than bottom

Figure 6: Checkerboard data as input: each rectangular feature is smoothly merged to a neighbour. Subfigures show: (a) a stack of horizontal slices, (b) taking a non-horizontal slice leads to a 'mixed- scale' map and (c) one mixed scale slice (non-horizontal plane).

The mixed-scale representation can also be obtained by slicing surfaces that are non-planar; e.g. a bellshape surface that could be used to create a meaningful 'fish-eye' type of visualizations (see Figure 7). This should be investigated with respect to the planar partition characteristic of the resulting maps. Probably OK in most situations, but it might be true that a single area object, in 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? Folding back surfaces seam to be non-sense as this will give two representations of the same object on same real world location in one map/visualization.



(a) (b) (c)

Figure 7: 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 surface) 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 level of detail the objects have, in such a way that the optimal number of objects is displayed; i.e. optimal information density. 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 objects, i.e. faces and edges (with certain number of coordinates), to be retrieved on average remains constant per viewport (independent from which level of detail is retrieved) and thereby the transfer and processing times stay within limits.

The optimal number can be realised, because the generalisation procedures that create tGAP data incrementally lead to less and less data in the hierarchy, i.e. less data is stored near the top of the space-scale cube (SSC) and the extent of area objects near the top of the cube is considerably larger (with a limited number of coordinates in their boundaries) than at the bottom (with more coordinates in their boundaries). 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 to take a clip of data out of such a slice: when a user is zoomed out, the viewport of a user will lead to a big 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 8 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 space-scale cube):

1. A request to provide an initial map based on simple 2D spatial range overlap selection of the relevant 3D polyhedra representing the vario-scale 2D objects in the requested area *A1* for the requested scale *s1* as illustrated in Figure 8(a). Note that the number of selected objects may be relatively large, so it can take some time before a map covering a requested area *A1* can be created by the client.
2. A request to provide an initial map based on overlap with a simple 3D block, i.e. a (orthogonal) spatial-scale range, overlap selection of the relevant 3D polyhedra representing the vario-scale 2D objects starting from the smallest scale *sn* (most coarse representation) until the required scale *s1* as illustrated in Figure 8(b). The server sends the selected 3D polyhedra sorted on smallest scale value, which enables progressive transfer for an area *A1*. The client can quickly start drawing an initial course representation, while still receiving additional detail.
3. A request to provide the 3D polyhedra for a progressive zoom-in as shown in Figure 8(c). Note the shrinking of the spatial selection range from an area *A1* at scale *s1* to an area *A0* at scale *s0* (a truncated pyramid up-side-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 8).
4. A request to provide the 3D polyhedra for a progressive zoom-out as shown in Figure 8(d). Note the growing of the spatial selection range from an area *A1* at scale *s1* to an area *A2* at scale *s2*. In this case the 3D polyhedra are sorted based on largest scale value from the larger to the smaller scale without sending 'intermediate' 3D polyhedra (again, not depicted in Figure 8).
5. A request to provide the 3D polyhedra for a simple pan operation from a first area *A1* to an adjacent area *A3* represented at the same scale *s1* as shown in Figure 8(e). In that case

the server immediately transmits the object data for the new map at the required level of detail.

6. A request to provide the 3D polyhedra for a progressive transfer pan as shown in Figure 8(f) from a first area *A1* to a second area *A3*. In that case the server subsequently transmits more and more detailed data (gradually changing from scale *sn* to scale *s1*) for the requested spatial range *A3*, and the client can gradually increase the level of detail with which the image data in said spatial range *A2* is displayed.

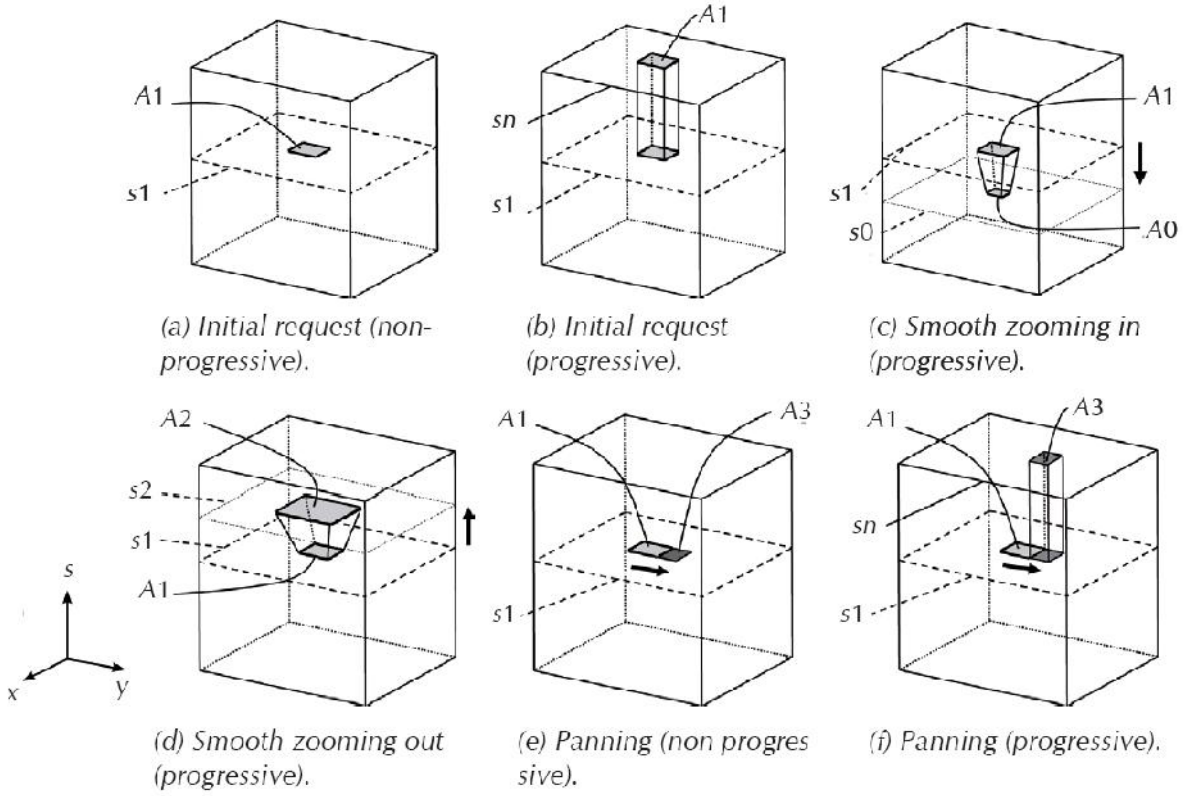


Figure 8. Zooming & panning with vario-scale data explained with the SSC (after van Oosterom and Meijers, 2011a).

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 height of the slice in the cube, together with taking the clip, should lead to a constant number of objects to be visualised. To realise the position of the cross section means that the question to be answered is 'which importance value 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 level of detail as the server can translate this extent to a suitable importance value to query the data structures. Make note that this on average is the right amount of information, as there may be regions with more or with less dense content than on average; e.g. rural vs. urban area.

4 CREATING THE STRUCTURE AND PROOF OF CORRECTNESS

This section discusses the question whether prism resulting split and merge operations (horizontal and vertical faces) can always be transformed into their smooth counterparts with non-horizontal faces? We provide the (intuitive) proof that the smooth tGAP structure can be created in all situations: including non-convex areas and areas with holes. 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 related to a simplified line by a set of tilted faces will not 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 (also error-free at the intermediate scales). This is because the new, tilted faces, are always moving within the 'free space' and can not 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. Proof: it is possible for strictly convex parts[2] of the disappearing area to be 'processed', using the following algorithm (see Figure 9):

- Count the number of interior nodes on the boundary to be removed (the so called 'shared boundary') and on the boundary to be moved to (the so called 'opposite boundary'). Note that at least one of these boundaries has the minimum number of 1 intermediate node, otherwise the neighbour to be removed would have no area.
- If unequal, add the missing number of (fake) nodes fairly distributed to the boundary with the too low number. The number of intermediate nodes is called *I(>0)* and is equal in both boundaries.
- Now that both boundaries have an equal number of nodes, add edges between each pair of corresponding intermediate nodes (so at least one edge is added).
- This results in two faces with three nodes and *I – 1* faces with four nodes (and also four) edges. If such a quadrangle face is not flat in 3D space, then add an additional diagonal edge and the resulting two triangles will be per definition flat.

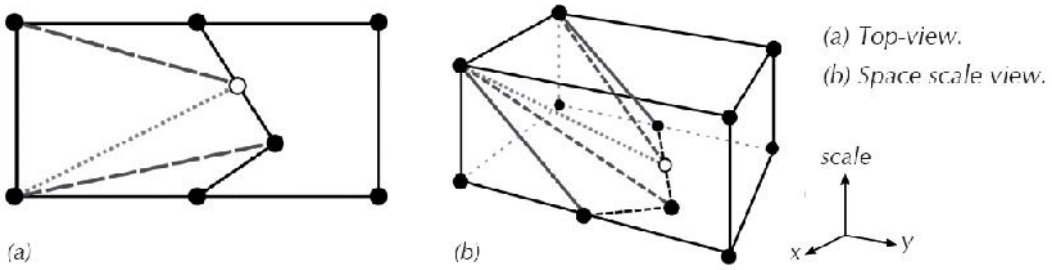


Figure 9: The simple neighbour merge: one rectangular feature is smoothly merged into rectangular neighbour feature. Note that the plane that forms the boundary between the two features is composed of 2 triangular and 1 quadrilateral faces (these faces can be dissolved by post-processing into 1 face, as they are planar).

[2] A simple polygon is strictly convex if every internal angle is strictly less than 180 degrees (so not equal to 180 degrees).

Note that this 'simple' algorithm may add some unneeded (temporary) nodes. Imagine two equal shaped neighbour rectangles, then a single diagonal face is sufficient. However, our algorithm would add two intermediate nodes (on the shared boundary) and create two triangles and one 4-node face. In a planarity check it may be detected that these faces are co-planar and can be merged (and same for split edges and added node may be removed). So, the final result is equal after this post-processing.

Because of the convex shape, there will never be intersecting edges or faces. If the to be merged shape 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 10(a). 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 10(b). Furthermore, this approach will also work as post-processing of a split operation: the split operation delivers boundaries to move to. Each part of a split polygon can, following the sketched recipe, result in a gradual merge with its neighbour.

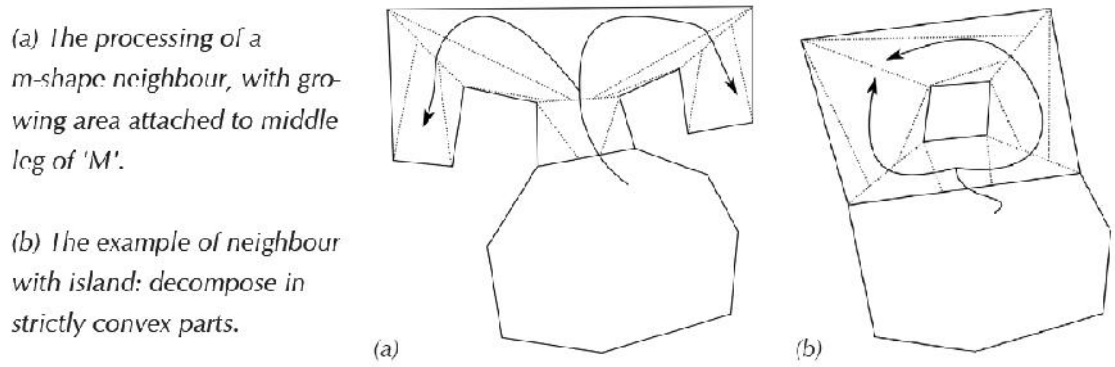


Figure 10. The processing of complex shapes into vario-scale representations. Note that quadrilaterals will not be planar and will have to be decomposed into triangular faces by adding an extra diagional.

A second approach, which does not require the 1-to-1 connection between intermediate nodes (but still requires convex parts) is as follows and illustrated in Figure 11: Count the number of segments in the shared *(n1)* and the opposite boundary *(n2)*. Now there will be n1 triangles constructed, which will have their base in the shared boundary (and the remaining vertex on the opposite boundary) and *n2 – 2* triangles which will have their base in the opposite boundary (and the remaining vertex on the shared boundary). This approach does not require post-processing. 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.

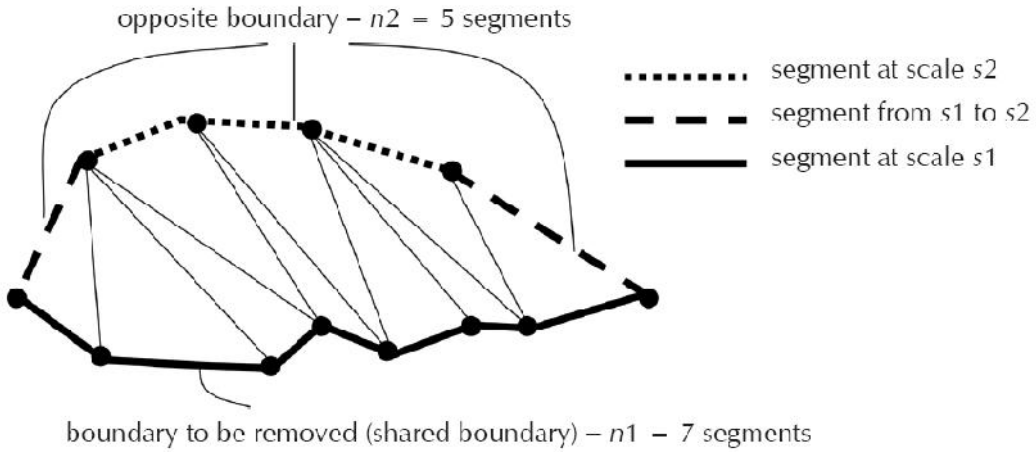


Figure 11. Alternative way of constructing non-horizontal faces. In this example 7 triangles have their base in the shared boundary and 5 – 2 = 3 triangles have their base in the opposite boundary.

5 THE 3D SMOOTH tGAP STRUCTURE

Until now a 2D base map was used. It is also possible to start with a 3D base map (model) and then create in a similar manner a 4D 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 (Level of Detail). In subsection 5.2 it will be explored how a 3D mixed scale representation can be obtained by using non-horizontal slicing hyperplanes.

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 12(a) 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 7 faces, because the left face has 2 parts: one of which is shared with object *zi*. These faces form the boundary segments.

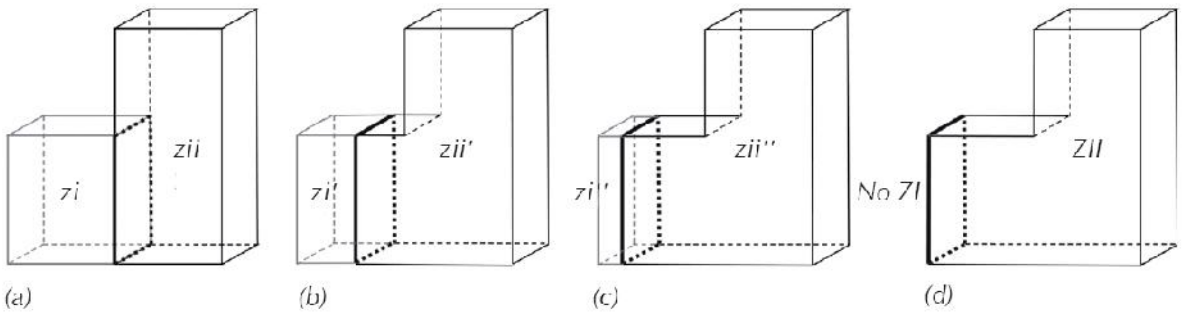


Figure 12. A simple 3D scene: the more important object *zii* gradually takes over the lesser important object *zi*.

Figure 12(a) shows a set of two 3D objects having the highest level of detail. Object *zi* represents for example a first building, object *zii* represents a second building. Also a lower level of detail 3D object representation is generated as shown in Figure 12(d). In this lower detailed 3D object-representation the original objects *zi* and *zii* are merged into object *ZII*. Figure 12(b) 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, Figures 13(a)-(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*.

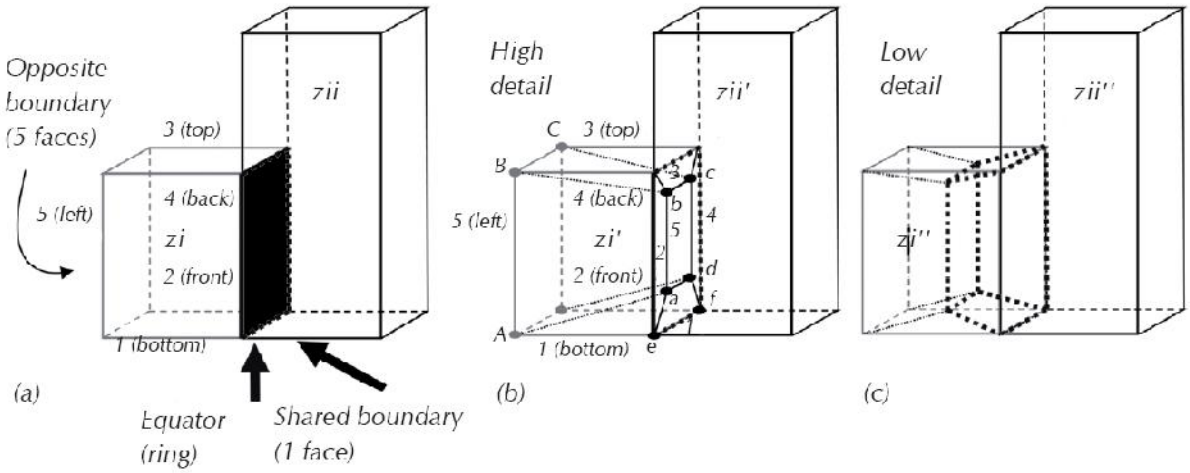


Figure 13. Alternative manner for gradually taking over the space of the lesser important object by the more important object.

Figure 13(a) again shows the higher detailed 3D object-presentation as in Figure 12(a). Objects *zi*, *zii* are convex objects[3] 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 12(d), this shared boundary is mapped to a destination boundary comprising bottom face 1, top face 3, front face 2, back face 4 and left side face 5 as indicated in Figure 13(a). As shown in Figure 13(b), 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,* etcetera mapped to *A – B, B – C*, etcetera, and faces 1 to 5 that are mapped to faces 1 to 5 of the destination (opposite) boundary. Lower and higher case characters indicate elements in the higher-detailed 3D representation and in the lower detailed 3D-representation respectively.

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

[3] In case of a concave object, this is then first decomposed into its convex parts, similar to the 2D approach.

two 3D representations at fixed scales *s1* and *s2* form the first two boundaries of the 4D SSC 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 1 to 5 that are mapped to faces 1 to 5 of the destination (opposite) boundary as indicated above. Each pair of a face 1 to 5 and its corresponding face 1 to 5 to which it is mapped delimits a trans-scale n-dimensional boundary segment in (*n+1*)-dimensional space. For example consider face 1 in the shared boundary, which is defined by nodes *a, d, f, e* in the higher detailed representation. This face 1 is mapped to corresponding face 1 in the destination boundary defined by *A, D, F, E* in the lower detailed representation. For clarity labels *E, 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 1: *a, d, e, f* in the higher detailed representation and its corresponding face 1: *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 2 to 5 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 an 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 level of detail that is position dependent; e.g. to support the generation of 3D perspective views (non-horizontal slicing hyperplanes); see Subsection 5.2.

Figure 13(c) shows an example of an intermediate representation obtained, in the generic general transition, when the slicing object has a value *s* equal to *(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, compare to the simple approach of Figure 12(b) 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 14(a)-(c) show examples of a simplify operation in the 3D case. Figure 14(a) shows a merged object *ZII* obtained by a merging operation as illustrated with reference to Figure 12(a)-(d) and with reference to Figure 13(a)-(c). Figure 14(b) shows a result according to a simplify operation of a first type, wherein the object is approximated by its 'bounding box', i.e. the smallest cuboid that contains the object. Figure 14(c) 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.

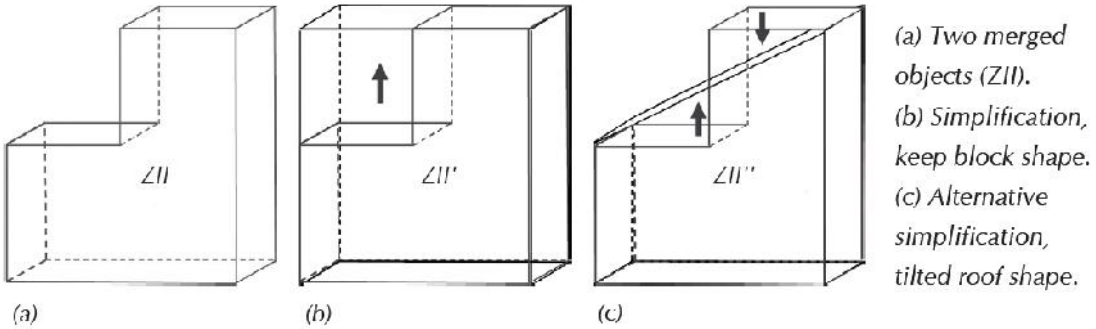


Figure 14. Two alternative options for the boundary simplification of the object *ZII*.

Figure 15(a) and (b) show a pseudo 4D impression for the merge followed by one of the two simplify 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, s2* for the additional scale dimension respectively. Figure 15(a) shows how the higher detailed 3D representation for object *zii* is assigned scale value *s1* in the 4D object-representation and how 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 15(b) shows how the higher detailed 3D representation for object *zii* is assigned scale value *s1* in the 4D objectrepresentation and how the lower detailed 3D representation for the resulting merged and simplified object *ZII''* (second type simplification) is assigned scale value *s2* in the 4D object-representation. The dotted lines indicate the trans-scale boundary segments (3D primitives in 4D space) between mutually corresponding nodes in the representations for *s1* and *s2*.

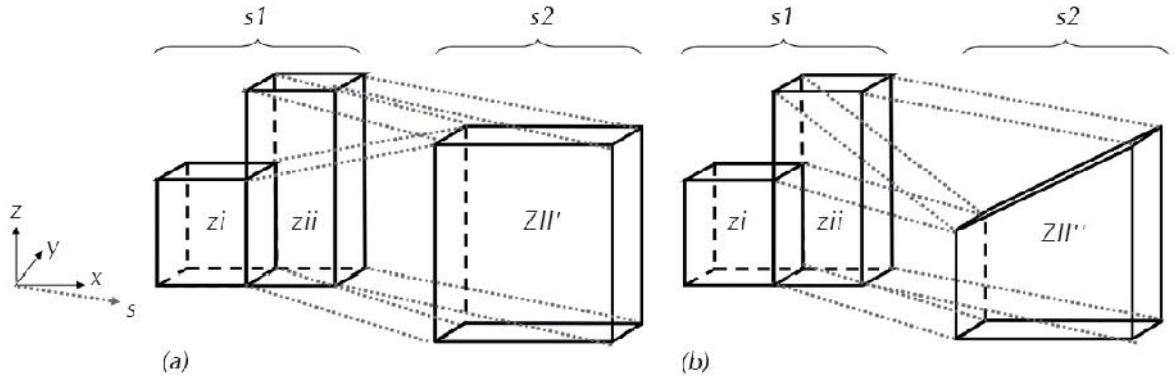


Figure 15. A pseudo 4D illustration of our simple 3D scene with two alternative vario-scale representations. Object *zi* is only shown for reference purposes. The grey lines llustrate the vario-scale representation of the second object; *zii* is gradually changed into *ZII',* respectively *ZII''* (depending on which simplification option is chosen).

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

The result 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 levels of details (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 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 (low in scale) far away from the viewer not so much detail (high in 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 Level of Detail (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 on-going 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 (and smaller scale deltas lead to smaller map deltas until and including the infinitesimal small delta) for all scales. The smoothness is accomplished by removing all horizontal faces of the classic tGAP structure. Recipes were given how to obtain data for 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. It is one integrated scale-space partition, and when using non-horizontal slices the resulting 2D maps will be a valid, mixed-scale planar partition: this is useful for use in 3D computer graphics. 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: how to encode the space-scale (hyper) cube in an efficient manner? Also create metrics and collect statistics: how many nodes, edges, faces, and volumes in the space-scale cube (and which primitives and references explicitly stored, c.f. van Oosterom et al., 2002; Meijers et al., 2009). An important decision is to encode the SSC with or without a topology structure. As the SSC is a full partition of the space-scale cube, it seems attractive to use a topology encoding because this allows efficient navigation through the data, avoid redundancy (double storage), guarantees consistent data (no gaps or overlaps), and fits very well to the tGAP structure.
- Formalize the structure and proof 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).
- Implementation of the smooth tGAP structure takes two main steps: 1. build classic tGAP and 2. transform from classic to smooth tGAP (space-scale cube). 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 map needs of the end-users).
- Testing 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). Probably different devices/platforms (desktop, mobile) have to be tested and users have to be given a range of relevant tasks. Large datasets result in large cubes, a slice near the bottom will contain

a lot of data (takes time) and is not what a user wants. So slicing, as explained in Section 3.2, should be combined with other (spatial) selection criteria; e.g. the bounding box (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.

- 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 1 by 1 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 1 by 1 sequencing is not implementing the steps in the structure in a sequential manner, but to group them and then let all members in the group transform in parallel. In van Oosterom and Meijers (2011b) some possible strategies to realize more parallel actions in the tGAP creation were proposed. These need further exploration.
- The smooth tGAP structure is mainly a DLM (digital landscape model) based representation, which is supporting application of cartographic styling at 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 a 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; e.g. 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; e.g. cylinder or sphere patches. One could also imagine the use of 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. 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 in part by the Dutch Technology Foundation STW (project numbers 11300 and 11185), which is part of the Netherlands Organisation for Scientific Research (NWO) and partly funded by the Ministry of Economic Affairs, Agriculture and Innovation. The authors would 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).

REFERENCES

- Baars, M., Stoter, J., van Oosterom, P., and Verbree, E. (2004). Rule-Based or Explicit Storage of Topology Structure: a Comparison Case Study. In Toppen, F. and Prastacos, P., editors, Proceedings of the 7th Conference on Geographic Information Science (CD-ROM), pages 765–769. Heraclion: Crete University Press.
- Been, K., Nöllenburg, M., Poon, S.-H., and Wolff, A. (2010). Optimizing active ranges for consistent dynamic map labeling. Computational Geometry, 43(3):312–328. Special Issue on 24th Annual Symposium on Computational Geometry (SoCG'08).
- Hägerstrand, T. (1970). What about people in regional science? Papers in Regional Science, 24(1):6–21.
- Hampe, M., Sester, M., and Harrie, L. (2004). Multiple representation databases to support visualization on mobile devices. In Proceedings of the XXth ISPRS Congress, volume XXXV of International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, pages 135–140, Istanbul, Turkey.
- 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(4):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(11):2191–2203. Progressive Transmission of Spatial Datasets in the Web Environment.
- Louwsma, J., Tijssen, T., and van Oosterom, P. (2003). Topology under the microscope. GeoConnexion.
- Meijers, M. (2011). Simultaneous & topologically-safe line simplification for a variable-scale planar partition. In Geertman, S., Reinhardt,W., and Toppen, F., editors, Advancing Geoinformation Science for a Changing World, Lecture Notes in Geoinformation and Cartography, pages 337–358. Springer Berlin Heidelberg.
- Meijers, M. and van Oosterom, P. (2011). The space-scale cube: An integrated model for 2D polygonal areas and scale. In 28th Urban Data Management Symposium, volume 38 of International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, pages 95–102.
- Meijers, M., van Oosterom, P., and Quak, W. (2009). A storage and transfer efficient data structure for variable scale vector data. In Advances in GIScience, Lecture Notes in Geoinformation and Cartography, pages 345–367. Springer Berlin Heidelberg.
- Noguera, J. M., Segura, R. J., Ogáyar, C. J., and Joan-Arinyo, R. (2010). Navigating large ter rains using commodity mobile devices. Computers & Geosciences, vol. 37, issue 9, pages 1218–1233.

- Nöllenburg, M., Merrick, D., Wolff, A., and Benkert, M. (2008). Morphing polylines: A step towards continuous generalization. Computers, Environment and Urban Systems, 32(4):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 Fendel, E. and Rumor, M., editors, Proceedings of UDMS'06 Aalborg, pages 9.61–9.71.
- Sester, M. and Brenner, C. (2005). Continuous generalization for visualization on small mobile devices. In Fisher, P., editor, Developments in Spatial Data Handling, pages 355–368. Springer-Verlag.
- Stoter, J., Meijers, M., van Oosterom, P., Grünreich, D., and Kraak, M.-J. (2010). Applying DLM and DCM concepts in a multiscale data environment. In Buttenfield, B., Brewer, C., Clarke, K., Finn, M., and Usery, L., editors, Proceedings of GDI 2010: Symposium on Generalization and Data Integration, pages 1–7, Boulder, USA. University of Colorado.
- Thompson, R. M. and van Oosterom, P. (2012). Modelling and validation of 3D cadastral objects. In Zlatanova, S., Ledoux, H., Fendel, E., and Rumor, M., editors, Urban and Regional Data Management - UDMS Annual 2011, pages 7–23, Leiden. CRC Press.
- van Kreveld, M. (2001). Smooth generalization for continuous zooming. In Proceedings 20th International Cartographic Conference (ICC'01), pages 2180–2185, Beijing, China.
- 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 Meijers, M. (2011a). Method and system for generating maps in an n-dimensional space. Dutch patent application 2006630, filed April 19, 2011, expected to be published October 2012.
- van Oosterom, P. and Meijers, M. (2011b). Towards a true vario-scale structure supporting smooth-zoom. In Proceedings of 14th ICA/ISPRS Workshop on Generalisation and Multiple Representation, pages 1–19, Paris.
- van Oosterom, P. and Stoter, J. (2010). 5D data modelling: full integration of 2D/3D space, time and scale dimensions. In Proceedings of the 6th international conference on Geographic information science, GIScience'10, pages 310–324, Berlin, Heidelberg. Springer-Verlag.
- van Oosterom, P., Stoter, J., Quak, W., and Zlatanova, S. (2002). The balance between geo- metry and topology. In Richardson, D. and van Oosterom, P., editors, Advances in Spatial Data Handling, 10th International Symposium on Spatial Data Handling, pages 121–135, Berlin. Springer-Verlag.
- Vermeij, M., van Oosterom, P., Quak, W., and Tijssen, T. (2003). Storing and using scale-less topological data efficiently in a client-server dbms environment. In GeoComputation 2003, University of Southampton, Southampton, UK.