Chapter 4
Data Structures for Continuous Generalisation: tGAP and SSC

Peter van Oosterom, Martijn Meijers, Jantien Stoter and Radan Šuba

Abstract Spatial zoom and thematic navigation are indispensable functionalities for digital web and mobile maps. Therefore, recent map generalisation research has introduced the first truly smooth vario-scale structure (after several near vario-scale representations), which supports continuous or smooth zooming. In the implementation, the vario-scale representation of 2D geo-information can be stored as a single 3D (2D+scale) data structure. A single uniform scale map in 2D is then derived by computing a horizontal slice through the structure.

4.1 Introduction

In recent years, the Internet has become an important source of digital maps for mobile users. However, applications suffer from bandwidth limitations and restricted devices such as small displays. Sending a map with many details for each request is expensive and time consuming. From a user's perspective this is unsatisfactory when zooming or panning, for example, when first navigating to an area of interest. As this task does not require a map at the highest resolution, it is reasonable to send a less detailed map first. In order to define these representations such that characteristic features are preserved, automatic generalisation methods are needed.

P. van Oosterom (✉) · M. Meijers · J. Stoter · R. Šuba
Section GIS Technology, Department OTB, Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, The Netherlands
e-mail: P.J.M.vanOosterom@TUDelft.nl

M. Meijers
e-mail: B.M.Meijers@TUDelft.nl

J. Stoter
e-mail: J.E.Stoter@TUDelft.nl

R. Šuba
e-mail: R.Suba@TUDelft.nl

D. Burghardt et al. (eds.), *Abstracting Geographic Information in a Data Rich World*, Lecture Notes in Geoinformation and Cartography, DOI: 10.1007/978-3-319-00203-3_4,

Technological advancements have led to maps being used virtually everywhere, for example map use in mobile smartphones. Map use is more interactive than ever before: users can zoom in, out and navigate via interactive maps. Therefore, recent map generalisation research has focused on continuous generalisation in contrast to 'traditional' generalisation applied to predefined scale-steps. Despite some useful efforts (Kreveld 2001; Sester and Brenner 2005; Cecconi and Galanda 2002), there is no optimal solution yet.

This chapter introduces the truly smooth 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 generalisations of real world features can be derived from a structure that can be used for presenting a smooth zoom action to the user. Furthermore, mixed-scale visualisations can be derived with more and less generalised features shown in one map in which the objects are consistent with each. Making such a transition area is one of the principal difficulties for 3D computer graphic solutions [e.g. using stitch strips based on triangles, such as used by Noguera et al. (2011)]. In addition, with the vario-scale approach there is no or minimal geometric data redundancy and there is no temporal delay between the availability of data sets at different map scales (as was and is the case with more traditional approaches).

Sections 4.2 and 4.3 provide the theoretical framework for vario-scale representations: the topological Generalised Area Partitioning-structure [tGAP, Sect. 4.2, largely based on van Oosterom (2005)] and the 3D SSC encoding of 2D vario-scale data [Space-Scale Cube, Sect. 4.3, largely based on van Oosterom and Meijers (2011b)]. After applying the theory to the two case studies in respectively Sect. 4.4 (constraint tGAP with independent constraint map), Sect. 4.5 (constraint tGAP with derived constraint map), the 3D approach of the SSC encoding of truly smooth tGAP is tested with Corine and ATKIS data in Sect. 4.6. Finally, Sect. 4.7 lists the main results and overview of future work.

4.2 Principles of the Generalised Area Partitioning Structure

This section presents the data structure for a variable scale representation of an area partitioning without redundancy of geometry, suitable for progressive transfer.

4.2.1 Introduction

There are a number of data structures available for multi-scale databases based on multiple representations, that is, the data are used for a fixed number of scale (or resolution) intervals. These multiple representation data structures attempt to explicitly relate objects at different scale levels, in order to offer consistency

during the use of the data. The drawbacks of the multiple representations data structures are that they store redundant data (same coordinates, originating from the same source) and they support only a limited number of scale intervals. Most data structures are intended to be used during the pan and zoom (in and out) operations, and in that sense multi-scale data structures are already a serious improvement for interactive use as they do speed-up interaction and give reasonable representations for a given level of detail (or scale).

Need for Progressive Data Transfer: A drawback of multiple representation data structures is that they are not suitable for progressive data transfer, because each scale interval requires its own independent graphic representation to be transferred. Good examples of progressive data transfer are raster images, which can be presented relatively quickly in a coarse manner and then refined as the user waits a little longer. These raster structures can be based on simple (raster data pyramid) (Samet 1984) or more advanced (wavelet compression) principles (Lazaridis and Mehrotra 2001; Hildebrandt et al. 2000; Rosenbaum and Schumann 2004). For example, JPEG2000 (wavelet based) allows both compression and progressive data transfer from the server to the end-user. Also, some of the proprietary formats such as ECW from ER Mapper and MrSID from LizardTech are very efficient raster compression formats based on wavelets, offering multi-resolution access suitable for progressive data transfer. Similar effects are more difficult to obtain with vector data and require more advanced data structures, though a number of attempts have been made to develop such structures (Bertolotto and Egenhofer 2001; Buttenfield 2002; Jones et al. 2000; Zhou et al. 2004).

Multi-scale/Variable-scale Vector Data Structures: For single (line) objects, a number of multi-scale/variable-scale data structures have been proposed: Strip-tree (Ballard 1981), Multi-Scale Line tree (Jones and Abraham 1987), Arc-tree (Gunther 1988), and the Binary Line Generalisation tree (BLG-tree) (van Oosterom 1990). The Strip-tree and the Arc-tree are intended for arbitrary curves, not for simple polylines. The Multi-Scale Line tree is intended for polylines, but it introduces a discrete number of detail levels and it is a multi-way tree, meaning that a node in the tree can have an arbitrary number of children. The BLG-tree is a binary tree for a variable-scale representation of polylines, based on the Douglas and Peucker (1973) line generalisation algorithm. Note that these line data structures cannot be used for spatial organisation (indexing, clustering) of multiple objects (as needed by variable-scale or multi-scale map representations), so they only solve part of the generalisation and storage problem.

One of the first multi-scale vector data structures designed to avoid redundancy was the reactive BSP-tree (van Oosterom 1989), which supports both spatial organisation (indexing) and multiple levels of details. Its main disadvantage, however, is that it is a static structure. The first dynamic vector data structure supporting spatial organisation of all map objects, as well as multiple scales, was the Reactive tree (van Oosterom 1992, 1994). The Reactive tree is an R-tree (Guttman 1984) extension with importance levels for objects: more important objects are stored higher in the tree structure, which makes more important objects more accessible. This is similar to the reactive BSP-tree, but the dynamic structure

of the Reactive tree enables inserts and deletes—functions that the BSP-tree lacks. The BLG-tree and the Reactive tree are eminently capable of supporting variable-scale/multi-scale maps composed of individual polyline or polygon objects.

Generalised Area Partitioning: The BLG-tree and Reactive-tree structures are not well suited for area partitioning, since removal of a polygon results in a gap in the map and independent generalisation of the boundaries of two neighbour areas results in small slivers (overlaps or gaps). Overcoming this deficiency was the motivation behind the development of the GAP tree (van Oosterom 1993). The BLG-tree, Reactive-tree, and GAP-tree data structures can be used together, while each supports different aspects of related generalisation processes, such as selection and simplification, or for area partitioning (van Oosterom and Schenkelaars 1995).

Following the conceptualisation of the GAP tree, several improvements were published to resolve limitations of the original data structures (van Putten and van Oosterom 1998; Ai and van Oosterom 2002; Vermeij et al. 2003). The next section describes the background of the topological GAP tree, which combines the use of the BLG-tree and the Reactive tree and avoids the problems of the original GAP tree—namely redundant storage and slivers near the boundary of two neighbouring areas.

4.2.2 GAP Tree Background

The first tree data structure for generalised area partitioning (GAP tree) was proposed by van Oosterom (1993). The idea was based on first drawing the larger and more important polygons (area objects), so as to create a generalised representation. However, one can continue by refining the scene through the additional drawing of the smaller and less important polygons on top of the existing polygons (based on the Painters algorithm; Fig. 4.1). This principle has been applied to the Digital Land Mass System-Digital Feature Analysis Data (DLMS DFAD) data structure (DMA 1986), because it already had this type of polygonal organisation. When tested with the Reactive tree and the BLG-tree, it was possible to zoom in and zoom out, and obtain map representations with more or less detail of a smaller/larger region in constant time (Fig. 4.2, left).

Computing the GAP Tree: If one has a normal area partition (and not DLMS DFAD data) one first has to compute the proper structure. This is driven by two functions. First, the importance function [for example: $Importance(a) = Area(a) * WeightClass(a)$] is used to find the least important feature a based on its size and the relative importance of the class it belongs to. Then the neighbour b is selected based on the highest value of $Collapse(a,b) = Length(a,b) * CompatibleClass(a,b)$, with $Length(a,b)$ being the length of the common boundary. Feature a is removed and feature b takes its space on the map. In the GAP tree this is represented by linking feature a as the child of parent b (and enlarging the original feature b). This process is repeated until only one feature is left covering the whole domain, forming the root of the GAP tree. Figure 4.1 gives a schematic representation of such a GAP tree.

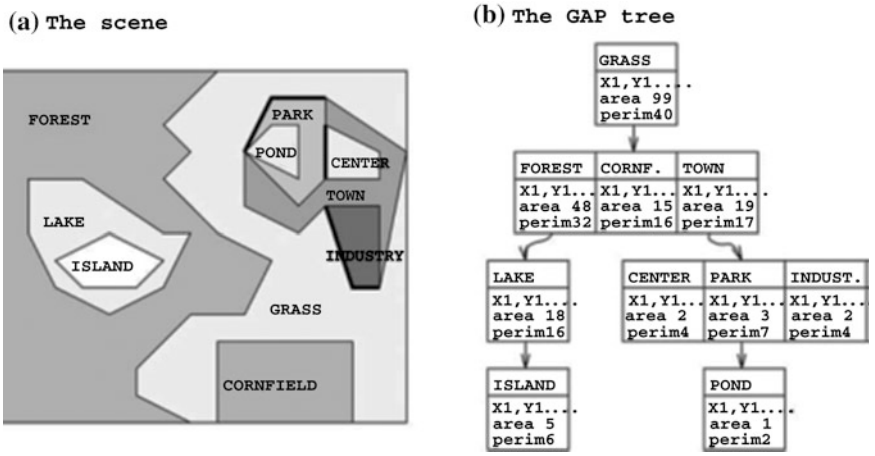

Fig. 4.1 The original GAP tree (van Oosterom 1993)

Work by van Smaalen (2003) focuses on finding neighbour patterns, which might in turn be used for setting up an initial compatibility matrix. Bregt and Bulens (1996) give area generalisation examples in the domain of soil maps, based on the same principles. Both van Smaalen (2003) and Bregt and Bulens (1996) use an adapted classification for the higher (merged) level of objects, instead of keeping the original classification at all levels of detail. For example, deciduous forest and coniferous forest objects are aggregated into a new object classified as "forest" or "garden," while "house" and "parking place" objects form the new object "lot". This could also be done in the GAP tree.

Implementations and Improvements of the GAP Tree: Though the GAP tree may be computed for a source data set, which has a planar partitioning topology, the GAP tree itself is not a topological structure. Each node in the GAP tree is a polygon, and this introduces some redundancy as parents and child may have some parts of their boundary in common. The first GAP-tree construction based on topologically structured input was implemented by van Putten and van Oosterom (1998) for two real world data sets: Top10vector (1:10,000) and GBKN (1:1,000; Fig. 4.2 right). It turned out that finding the proper importance and compatibility functions (which drive the GAP-tree construction) is far from trivial and depends on the purpose of the map. In addition, two improvements were presented in the 1998 paper (at the conceptual level): (1) adding parallel lines to "linear" area features, and (2) computing a GAP tree for a large seamless data set.

Ai and van Oosterom (2002) presented two other possible improvements to the GAP tree: One improvement was that the least important object should not only be assigned to one neighbour, but subdivided along its skeleton and the different parts assigned to different neighbours/parents (the result is not a tree but a directed acyclic graph: GAP-DAG). The second improvement concerned extending the neighbourhood analysis by considering non-direct (sharing a common edge)

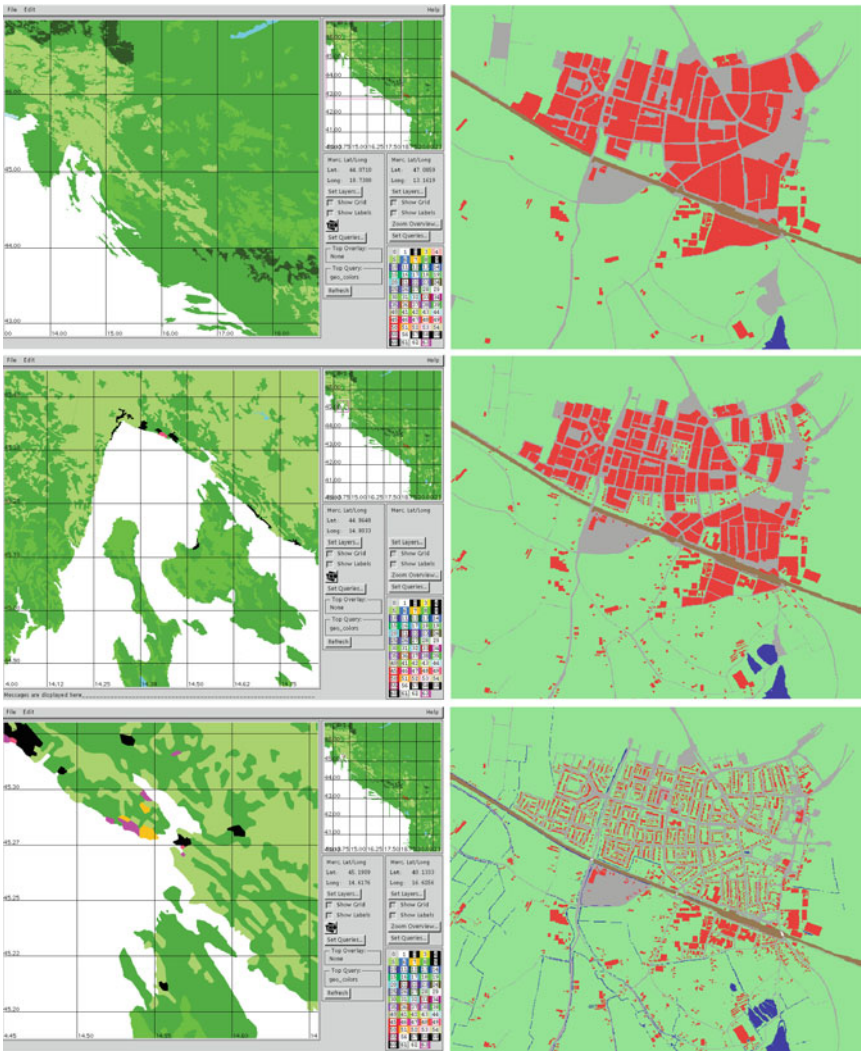

Fig. 4.2 *Left* GAP-tree principle applied to DLMS DFAD (adds detail when zooming in). *Right* GAP tree applied to large scale topographic data set (shown at the same scale)

neighbour areas as well. Both suggestions are based on an analysis using a Triangular Irregular Network (TIN) structure.

Topological Version of the GAP Tree: All improvements still result in a non-topological GAP structure, which means that it contains redundancy. Vermeij et al. (2003) presented a GAP-tree structure that avoids most redundancy by using a topological structure for the resulting GAP tree, not only for the input: thus the edges and the faces table both have attributes that specify the importance ranges in which a given instance is valid. The 2D geometry of the edges (and faces) is

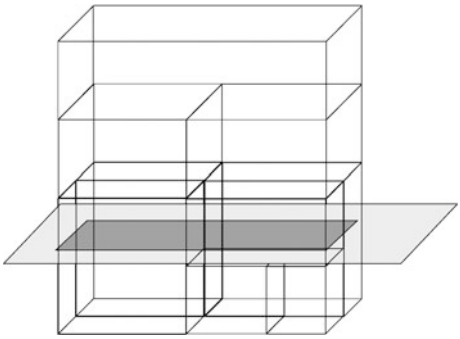

Fig. 4.3 Importance levels represented by the third dimension [at the most detailed level (*bottom*) there are several objects, while at the most coarse level (*top*) there is only one object]. The hatched plane represents a requested level of detail, and the intersection with the symbolic 3D volumes then gives the faces

extended by the importance value range (on the z-axis) for which it is valid (Fig. 4.3). One drawback of this approach is that it requires considerable geometric processing at the client side—clipping edges, forming rings, and linking outer and possible inner rings to a face. A second drawback is that there is some redundancy introduced via the edges at the different importance levels: i.e., the coordinates of detailed edges are again present in the edge at the higher aggregation level. Finally, van Oosterom (2005) introduced a data structure which also avoids the redundant storage of geometry at different levels of detail (in the edges). Figure 4.4 shows the generalisation process for the tGAP structure in which a simplified version of the edges is available, without explicitly storing them.

4.3 Space Scale Cube for Vario-Scale

This section introduces a truly smooth 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 generalisations 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 visualisations can be derived with more and less generalised features shown together in which the objects are consistent with each other. Making such a transition area is one of the principal difficulties for 3D computer graphic solutions (e.g. using stitch strips based on triangles, such as used by Noguera et al. (2011).

The remainder of this section is structured as follows: Sect. 4.3.1 contains a discussion on how the classic tGAP structure can be represented by a 3D space-scale cube and how this can be adapted to store a more continuous generalisation. The third dimension is used to encode the tGAP representation, resulting in a

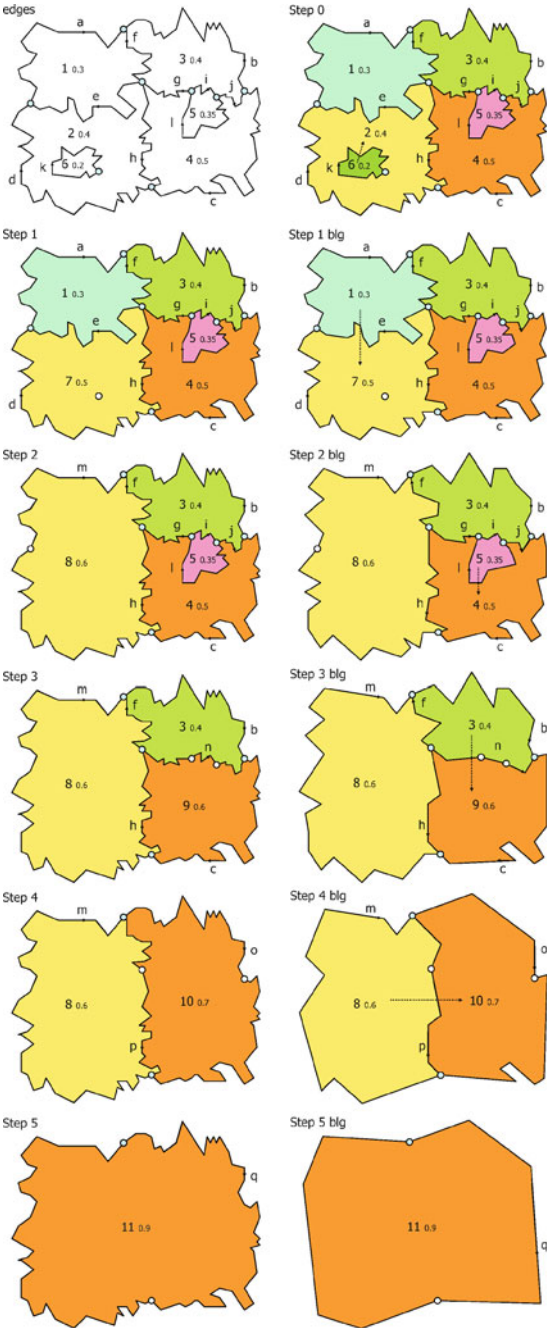

Fig. 4.4 Generalisation example in *five steps*, from detailed to course. The *left side* shows the effect of merging faces, while the *right side* shows the effect of also simplifying the boundaries via the BLG tree. Note that nodes are depicted in *green*/*blue* and removed nodes are shown for one next step only in *white*

volumetric partition of stacked prisms (polygons extruded according to their valid scale range). Maps are created by horizontal slices and continuous generalisation corresponds to gradually moving the slicing plan (and scaling). The classic encoding of the tGAP with stacked prisms, has horizontal faces causing sudden local shocks when moving up or down the slicing (zoom) plane. The truly smooth tGAP structure is obtained by removing the horizontal faces and replacing them with tilted faces. The support of additional generalisation operators in a SSC is described in Sect. 4.3.1. Section 4.3.2 illustrates how a mixed scale representation is obtained from the smooth tGAP structure by taking non-horizontal slices. The presented data structures and methods are valid for both 2D and 3D data.

4.3.1 The tGAP Structure Represented by the 3D Space-Scale Cube

The tGAP structure has been presented as a vario-scale structure (van Oosterom 2005). The tGAP structure can be seen as a result of the generalisation process and can be used efficiently to select a representation at any required level of detail (scale or height). Figure 4.5 shows four map fragments and the tGAP structure in which the following generalisation 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 neighbouring farmland;
3. Simplify boundary between farmland and water area.

The tGAP structure is a DAG and not a tree structure, as the split causes the road object to have several parents (Fig. 4.5e). 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 chapter, it is clearer to illustrate these operators separately. For the tGAP structure, the scale has been depicted in the 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 4.6a shows this 3D representation for the example scene of Fig. 4.5. 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 consists of a compound geometry with a 3D volume-part and 2D face-part attached.

There are many small steps from the most detailed to the most coarse representation—in the classic tGAP, $n - 1$ steps exist, if the base map contains n objects. Nevertheless, this could still be considered as many discrete generalisation actions approaching vario-scale, but not truly smooth vario-scale. Split and

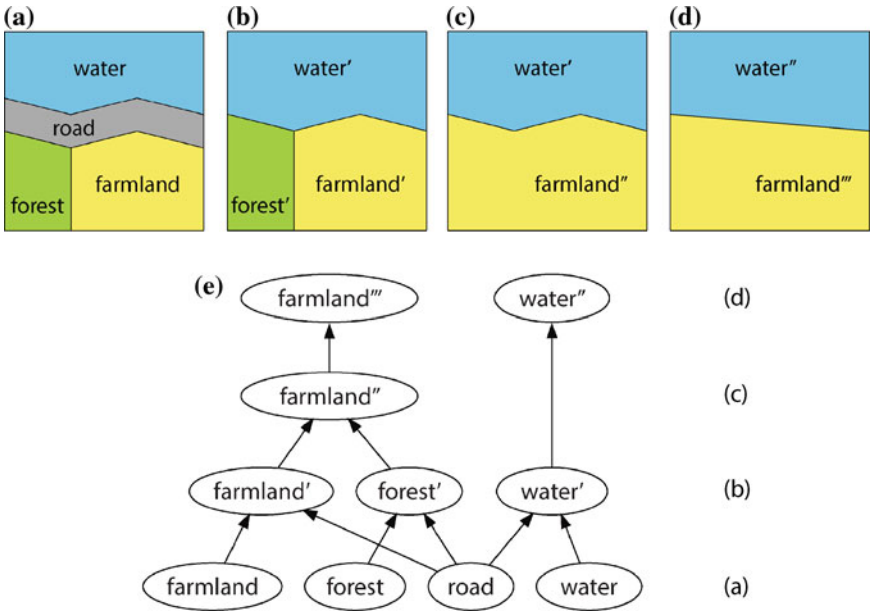

Fig. 4.5 The 4 map fragments and corresponding tGAP structure. **a** Original map. **b** Result of collapse. **c** Result of merge. **d** Result of simplify. **e** Corresponding tGAP structure

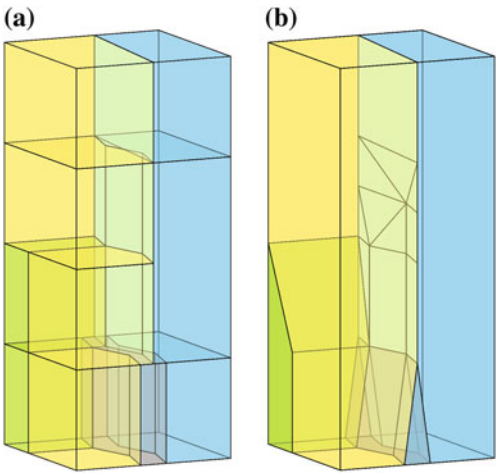

Fig. 4.6 The space-scale cube (*SSC*) representation in 3D: **a** SSC for the classic tGAP structure. **b** SSC for the smooth tGAP structure

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; (Fig. 4.7a, b). In the space-scale cube this is represented by a horizontal face; a sudden end or start of a corresponding object. Furthermore, polygon boundaries

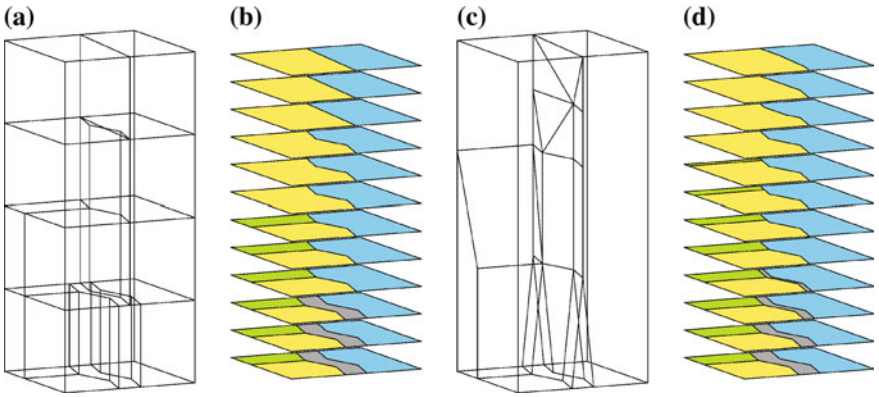

Fig. 4.7 Comparing the classic and smooth tGAP/SSC structures: **a** wireframe classic, **b** map slices classic, **c** wireframe smooth, and **d** map slices smooth. Note that nothing changes until a tGAP event has happened in classic structure (**a**–**b**)

define faces that are all vertical in the cube, i.e. the geometry does not change at all within the corresponding scale range, which result in prisms constituting a full partition of the space-scale cube. Replacing the horizontal and vertical faces with tilted faces as depicted in Fig. 4.6b, results in smooth transitions; (Fig. 4.7c, d).

4.3.2 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 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 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 lead to progressively less data in the hierarchy, i.e. less data are 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 to 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

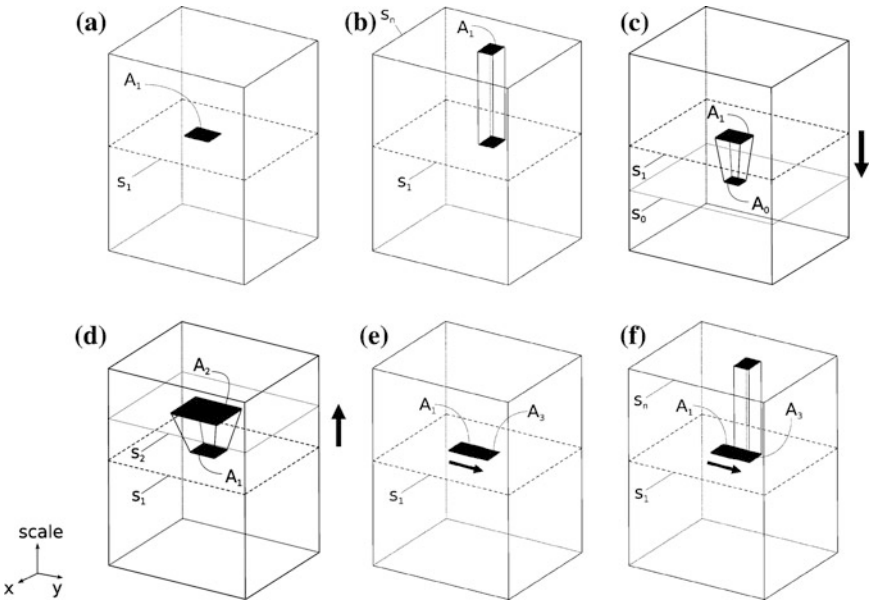

Fig. 4.8 Zooming and panning with vario-scale data explained via the SSC [after van Oosterom and Meijers (2011a)]: **a** Initial request (non-progressive). **b** Initial request (progressive). **c** Smooth zooming in (progressive). **d** Smooth zooming out (progressive). **e** Panning (non-progressive). **f** Panning (progressive)

move the extent of the clip within the horizontal slicing plane. Figure 4.8 illustrates this idea. 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 Fig. 4.8a. 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 2D 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 (Fig. 4.8b). 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 Fig. 4.8c. 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 Fig. 4.8).

4. A request to provide the 3D polyhedra for a progressive zoom-out as shown in Fig. 4.8d. 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 Fig. 4.8).
5. A request to provide the 3D polyhedra for a simple pan operation from a first area $A1$ to an adjacent area $A3$ represented a the same scale $s1$ as shown in Fig. 4.8e. 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 Fig. 4.8f from a first area $A1$ to a second area $A3$. This case is comparable to the case 2 'initial 2D map with progressive transfer' for the new part of requested spatial range $A3$. In this 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 displayed level of detail.

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 a certain direction (up or down) and may avoid repeating parts of the needed geometric computations. This is similar to plane-sweep algorithms 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 visualised. To realise the position of the cross section means that the question to be answered is 'which height corresponds to the map scale on the client side?' 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 it 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 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 areas. 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 while 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).

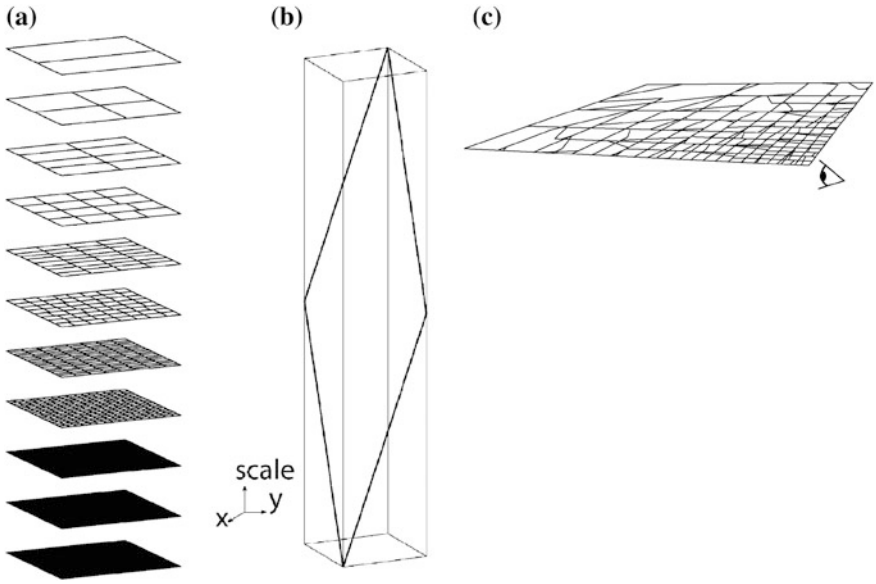

Fig. 4.9 Checker board data as input: each *rectangular* feature is smoothly merged to a neighbour. Subfigures show: **a** a stack of horizontal slices, **b** taking a non-horizontal slice leads to a 'mixed-scale' map and **c** one mixed scale slice (*non-horizontal plane*)

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

4.3.3 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 4.9 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 cube, less detail where the slice is closer to the top. Consider 3D visualisations, 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 generalised (small scale) features far away and less generalised (large scale) features close to the observer.

The mixed-scale representation can also be obtained by slicing surfaces that are non-planar; e.g. a bell-shaped surface that could be used to create a meaningful 'fish-eye' type of visualisations (Fig. 4.10). This is likely to be the desired

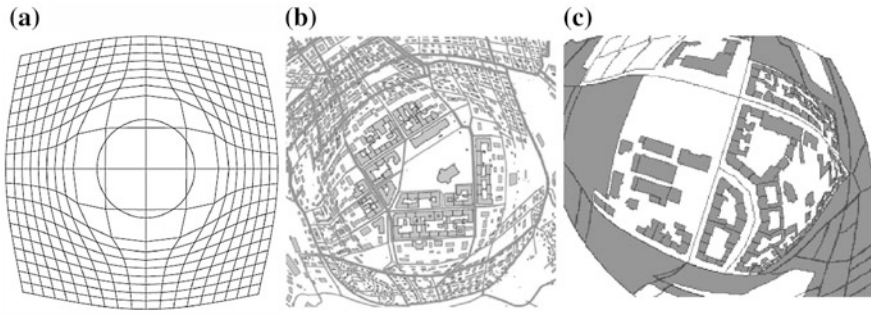

Fig. 4.10 A 'mixed-scale' map. Harrie et al. (2002) term this type of map a 'vario-scale' map, while we term this a 'mixed-scale' map. Furthermore it is clear that there is a need for extra generalisation closer to the borders of the map, which is not applied in (**b**), but is applied in (**c**). With our solution, this generalisation 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)

outcome 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 nonsensical as it could lead to two representations of the same object in one map/visualisation.

The artificial checker board '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 the SSC in Fig. 4.6a 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 very well, truly smooth zooming. More usability tests will be needed to evaluate the dynamic map types the users prefer with more realistic map data.

4.4 Case Study I: Dutch Large Scale Basic Topographic Data in Constraint tGAP[1]

The tGAP (topological Generalised Area Partitioning) structure is designed to store the results of the gradual generalisation process and to support smooth zoom as described in Sect. 4.2. In this section the cartographic quality of the tGAP is improved by using a known smaller scale high quality representation 'as target' when taking the

[1] Case Study I is a selection, by the chapter authors, from the publication (Dilo, van Oosterom, Hofman 2009).

generalisation decisions on the large scale input data when constructing the tGAP structure. With the additional smaller scale input data (the constraints), the iterative algorithm can be controlled to obtain higher quality (intermediate) maps.

The generalised dataset (smaller scale input) can be obtained via two routes: (1) from an external, independent source or (2) via a different generalisation algorithm. The first option is explored in this section where the idea was implemented with real topographic datasets from the Netherlands for the large- (1:1,000) and independent medium-scale (1:10,000) data. The second option is investigated in Sect. 4.5, where the small scale input is derived from the same base data (via an optimisation approach using mixed integer programming).

4.4.1 Topographic Datasets of Large and Medium Scale

Generalisation for the tGAP data structure is performed on an area partition; thus the large-scale dataset is required to be an area partition. Area objects of the smaller scale data set act as region constraints in the generalisation process, i.e. they restrict the aggregation of large-scale objects only inside the region constraints. The method proposed here consists of two stages: the first stage matches objects of the large-scale dataset to objects of the smaller scale dataset, which act as region constraints in the next stage; the second stage compiles additional information needed for the constrained tGAP and performs the generalisation.

In this study we use large- and medium-scale topographic data from the Netherlands, at 1:1,000 and 1:10,000 scales, respectively. Large-scale topographic data in the Netherlands are mostly produced by municipalities, while smaller scale maps are produced by the Netherlands' Kadaster (who also holds the mapping agency). IMGeo is the information model for large-scale data at the scale 1:1,000. Top10NL is the model for topographic data at the scale 1:10,000. The datasets in this study follow the two models, IMGeo and Top10NL. All the illustrations in this section were prepared with data from the region of Almere in the Netherlands.

The two models, IMGeo and Top10NL, both endorse this meaning of topography (relief is not included as we focus on the area partitioning). The main feature classes of IMGeo are Road, Railway, Water, Building, LayoutElement and Terrain. Class LayoutElement has 11 subclasses, each representing a different kind of topographic element, e.g. Bin, StreetFurniture and OtherBuilding. The class Terrain contains anything that is not buildings or infrastructure, for example, forest, grassland, fallow land, etc. Although the (area) objects of IMGeo should form an area partition, this was not the case for the test data. We processed the test data and created a partition. The overlaps were resolved by imposing an importance order on classes: layout elements have the highest importance, followed by roads and water, while terrain has the lowest importance. This order is translated to rules. For example, 'if a Terrain object overlaps with another object, subtract the overlap from the Terrain object'; see Hofman (2008) for a complete set of rules. The IMGeo test data allows categorisation mainly on classes. The class Terrain

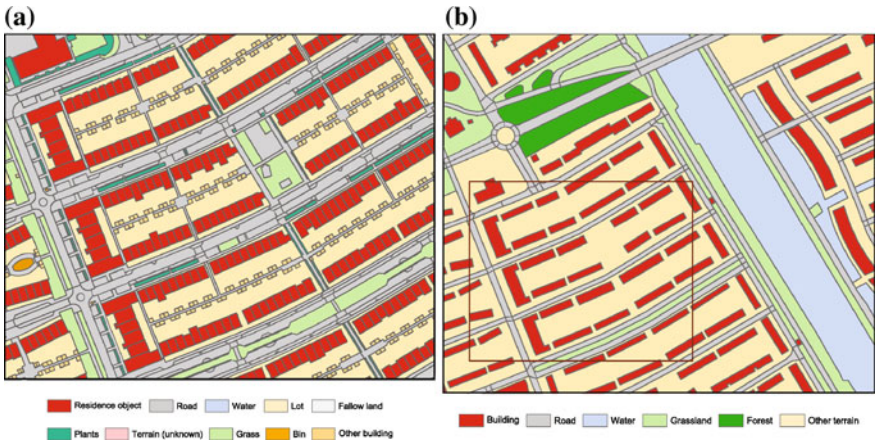

Fig. 4.11 **a** Large-scale topographic data from the city of Almere, the Netherlands. **b** Medium-scale topographic data from the city of Almere, the Netherlands. The *rectangle* in the *middle of the image* shows the extent of the map of (**a**)

can be further categorised by the LanduseType attribute values, and the class LayoutElement allows categorisation based on its subclasses. We created a new attribute class to store these categories: building; road; water; lot, fallow land, plants, other terrain, and grass, for the land-use values of Terrain; bin, and other buildings as subclasses of the LayoutElement class. Figure 4.11 shows the IMGeo data for part of the city of Almere, the Netherlands.

The Top10NL model contains all the IMGeo classes, which have similar attributes to their corresponding classes in IMGeo. We would expect the topographic model of a smaller scale to be less detailed than the model of a larger scale. This is not always the case for Top10NL compared to IMGeo, e.g., Top10NL differentiates between different kinds of forests, while IMGeo classifies them all as forest. The Top10NL objects were categorised based on the main classes. Additionally, objects of class Terrain were further categorised according to the LanduseType attribute values. A new attribute, region-class, was created for the Top10NL data, with the following values: building; road; water; grassland, forest, and other terrain from the land-use values of Terrain. Terrain objects had overlap with objects of other classes. There were also overlaps between road and water objects. We also processed the Top10NL data using the importance order of classes, and created a partition. Figure 4.11b shows Top10NL data from the city of Almere, covering a larger area than Fig. 4.11a. The rectangle in the middle of the map shows the extent of the map in Fig. 4.11a.

Some of the categories have the same semantics in both models, e.g. Road and Water. Some categories have similar semantics, e.g. Building in IMGeo is a residence unit, while in Top10NL it is a building block, Grass in IMGeo is similar to Grassland in Top10NL. There are categories of IMGeo that do not have a one-to-one match to the categories of Top10NL. For example, Other building from

Fig. 4.12 Large-scale IMGeo building objects (in *red*) overlaid with medium-scale Top10NL buildings blocks (in *blue*, semi-transparent). The background is the large-scale dataset in *dimmed colours*. It should be noted that for other feature classes the matching is often also very challenging

IMGeo can be a Building or Other terrain in Top10NL, while Other terrain of Top10NL can be Lot or Fallow land in IMGeo. Semantic matching is considered in the object matching described in the next section.

4.4.2 Data Pre-Processing and Object Matching

The constrained tGAP has some assumptions, which put requirements on the data we want to generalise. The first assumption is that data has to form an area partition, which was not the case for IMGeo data of Almere and Rotterdam as overlapping areas did occur, which had to be corrected. The second assumption is that we know in advance to which region (constraint) each object of the original data belongs. We want to use Top10NL objects as region constraints for the IMGeo data, while the two are created independently. Figure 4.12 shows overlay of Top10NL and IMGeo data, illustrating the matching problem. We need to assign a Top10NL object to every IMGeo object before we can run the constrained tGAP code. After pre-processing, IMGeo and Top10NL, each have become a partition. To assign IMGeo objects to region constraints, i.e. Top10NL objects, several methods were investigated and implemented. Four possible methods to assign IMGeo objects to Top10NL regions were developed:

1. The *simple overlay* method: An intersection between the models where every IMGeo object is split at the borders of the overlapping Top10NL region. In the end result only Top10NL geometry will be visible.

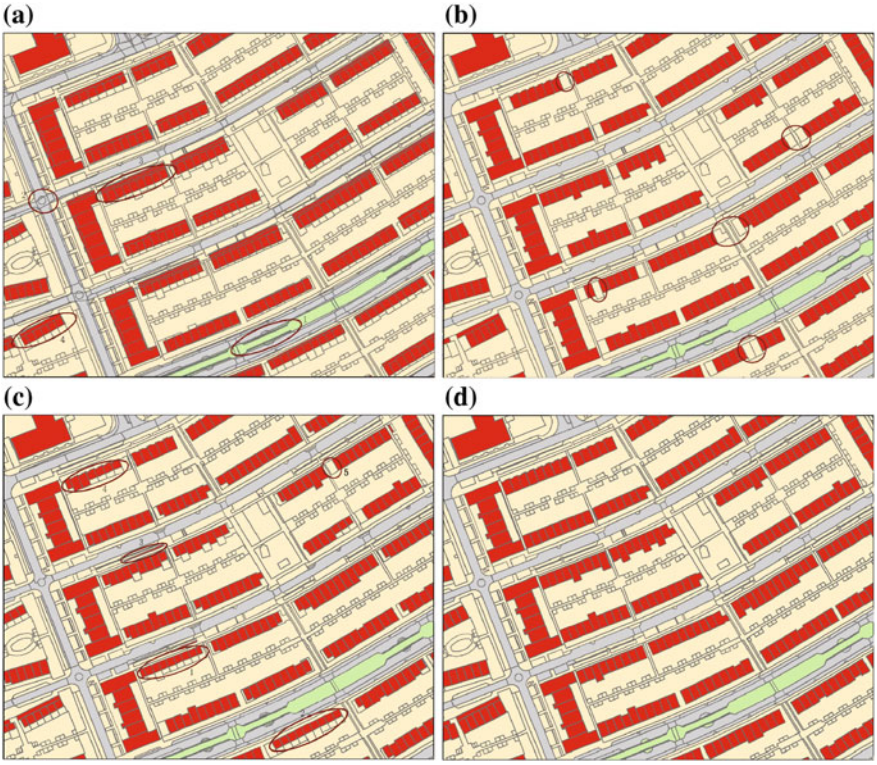

Fig. 4.13 The results of the four object matching methods visualised: **a** Simple overlay (note slivers). **b** Maximum Area method (note missing building in some blocks). **c** Split method (note that some blocks have been deformed by slit 'don't understand what you mean by slit'). **d** Building First method (least negative side effects). The IMGeo objects are coloured according to the region they belong to, which gives an indication of the constrained tGAP end result

2. The *maximum area* method: The Top10NL region which overlaps the IMGeo object the most is the shape to which the whole IMGeo object is assigned to. The IMGeo geometry is kept in this method.
3. The *35 %-split* method: If an IMGeo object belongs for more than 35 % to two Top10NL regions we consider this Top10NL geometry as enrichment of the structure; therefore the IMGeo object is split and new IMGeo objects are created. For all other IMGeo objects the maximum area method is applied.
4. The *building first* method: This method assigns IMGeo-buildings to a building region in the case where there is some overlap with a Top10NL building block, without considering the amount of overlap. The other IMGeo objects are processed as in the maximum area method.

As the last method proved to give the best matching results (based on our visual inspection of applying the various methods to our test data (Fig. 4.13), we decided to continue with matching results using this method.

Table 4.1 Class weights determined for the vario-scale IMGeo and compatibility values for the vario-scale IMGeo

From-class code →		1001	2001	3001	4001	4002	4003	4004	4005	5001
Weight		**13.0**	**1.20**	**1.30**	**9.00**	**1.00**	**0.93**	**0.90**	**0.88**	**1.00**
Class name	To-c↓									
Building	1001	1.00	0.50	0.00	0.90	0.90	0.50	0.50	0.00	0.99
Road	2001	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Water	3001	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Lot	4001	0.50	0.90	0.00	1.00	0.95	0.95	0.95	0.80	0.95
Fallow land	4002	0.90	0.90	0.00	0.50	1.00	0.90	0.90	0.50	0.95
Plants	4003	0.50	0.50	0.00	0.50	0.50	0.99	0.95	0.90	0.50
Terrain unk.	4004	0.90	0.50	0.00	0.50	0.90	0.00	1.00	0.99	0.95
Grass	4005	0.50	0.90	0.00	0.80	0.50	0.95	0.95	0.99	0.50
Other Building	5001	0.99	0.50	0.00	0.50	0.90	0.50	0.50	0.00	1.00

Motivation was readability as Table 4.1 shows both class weight (in bold) and class compatibility matrix

4.4.3 Constrained tGAP Generalisation

The generalisation process is performed in steps very similar to the normal tGAP. In each step, the least important object is merged to its most compatible neighbour, forming a new object. This pair wise merging is controlled by region constraints: *objects are allowed to merge only if they belong to the same smaller scale region*. Inside the constraints, the generalisation results are driven by the importance and compatibility values of objects. The importance value of an object v is calculated from the area size, and the weight of the object's category: $Imp(v) = area\_size(v) * weight(class(v))$. The compatibility between two objects u and v is calculated from the length of the shared boundary, and the compatibility values between their categories: $Comp(u,v) = bnd\_length(u,v) * compatib(class(u), class(v))$. The generalisation stops when all the objects are merged up to the region constraints. The initial values for weights and compatibilities were based on the work of van Putten and van Oosterom (1998). Next, these values were tuned in order to get better generalisation results. The tuning of values was by trial and error, based on visual inspection of the results, with the aim of a gradual transformation of the large-scale dataset toward the medium-scale dataset. The optimum values found for the weights (line in bold) and the compatibilities are given in Table 4.1. Buildings are important and should be retained through all the scales of the maps. We ensure this by granting a high weight value to the category Building. The IMGeo dataset of Almere considers parking places and sidewalks as roads, while the Top10NL dataset does not. This results in more road objects in the IMGeo data as compared to the Top10NL data. To achieve a gradual decrease of roads during the generalisation, we attach a relatively small weight to roads, as well as a very low compatibility of other categories to roads (for row 2001, most of the values are 0, Table 4.1) while allowing relatively high compatibilities of roads

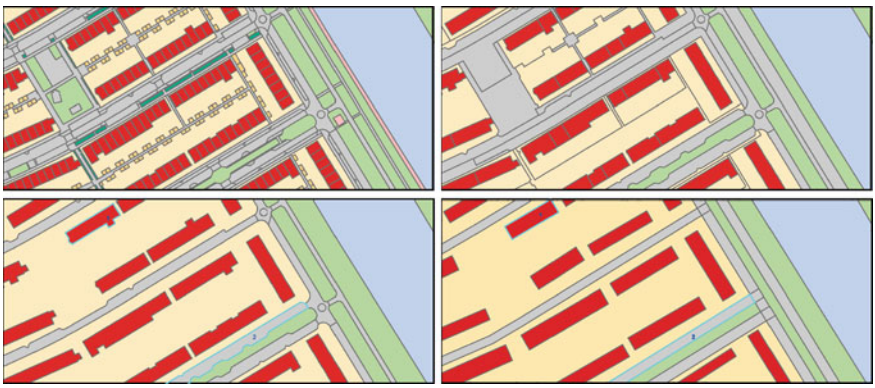

Fig. 4.14 Results constrained tGAP: (*top left*) original IMGeo 1:1,000, (*top right*) result scale 1:5,000, (*bottom left*) result scale 1:10,000, (*bottom right*) Top10NL 1:10,000

to other categories (see column 2001). We consider water to be compatible only with itself, and incompatible with other categories: compatibility of water to any other category equals 0, and also compatibility of any other category to water equals 0. The IMGeo categories, lot, fallow land, terrain (unknown) and other building have a similar semantics with the category terrain of Top10NL, thus we want the objects of these categories to aggregate together. We make Lot an attractive category by granting it a high weight value and high compatibilities of the other categories to lots, while allowing it to merge to the other categories, which is ensured by the medium–high compatibility values of lots to the other categories.

The tGAP structures are stored as Oracle tables, and the constrained tGAP generalisation is performed by code written in PL/SQL. Oracle tables store data for the constrained tGAP, their relations, as well as primary and foreign keys. Region constraints subdivide the whole domain into parts that can be processed independently and therefore in parallel. In addition, due to this the tGAP operations are local (i.e. look for least important feature within region constraint in contrast to looking for globally least important feature). This is relevant when considering updates: the original tGAP iteratively looks for the globally least important feature, and a small change in the largest may cause quite a different resulting structure. This is not the case for the constrained tGAP, where the effect of a change is limited to the area of the involved region constraint.

4.4.4 Constrained tGAP Generalisation: Conclusions

The maps of Fig. 4.14 illustrate results of the generalisation using weight and compatibility values as discussed in the previous section. The maps show the same part of Almere, for comparison: the original IMGeo data (scale 1:1,000) in top left

map, the result of the constrained tGAP generalisation for the scale 1:5,000 in top right map, the result of the constrained tGAP generalisation for the scale 1:10,000 in bottom left map, and the original Top10NL data (scale 1:10,000) in bottom right map. Comparing tGAP generalisation for the scale 1:10,000 (middle-bottom map), the result of this research, with the Top10NL 1:10,000 dataset (bottom map), it can be seen that still some line simplification is needed to further improve the results (for example, compare the boundary lines of the selected objects: the building labelled 1 and the road labelled 2). However, it can also be concluded that the results are a good generalisations of the original data. More tests with the constrained tGAP generalisation applied to Rotterdam data support this conclusion [see Hofman (2008) for more details].

4.5 Case Study II: German Land Cover Data in Constraint tGAP[2]

Instead of using large scale topographic data the second case is using land cover data from ATKIS DLM50, which starts at the medium scale (1:50,000). There is no smaller scale available as a constraint. However, via an optimisation procedure a 'one-step/one-scale' smaller scale representation (1:250,000) is derived. This derived representation is then used as constraint for the vario-scale structure. The advantage of this derived smaller scale over an independent smaller scale map, is that there are rarely any mismatches between the larger scale objects.

In order to create a better tGAP-generalised dataset (without having a second medium or smaller scale dataset available as a constraint), we propose a method of two stages: First, we create a generalised representation from a detailed dataset, using an optimisation approach that satisfies certain cartographic rules (e.g. minimum size for objects that are shown on the map). Secondly, we define a sequence of basic tGAP merge and simplification operations that transforms the most detailed dataset gradually into the generalised dataset. The obtained sequence of gradual transformations is stored without geometrical redundancy in a structure that builds up on the previously developed tGAP (topological Generalised Area Partitioning) structure. This structure and the algorithm for intermediate levels of detail (LoD) have been implemented in an object-relational database and tested for land cover data from the official German topographic dataset ATKIS at a scale of 1:50,000 to the target scale 1:250,000. Results of these tests allow us to conclude that the data at the lowest LoD and at intermediate LoDs is well generalised. Applying specialised heuristics the applied optimisation method can cope with large datasets; the tGAP structure allows users to efficiently query and retrieve a dataset at a specified LoD.

[2] Case Study II is a selection, by the chapter authors, from the publication (Haunert, Dilo, van Oosterom 2009).

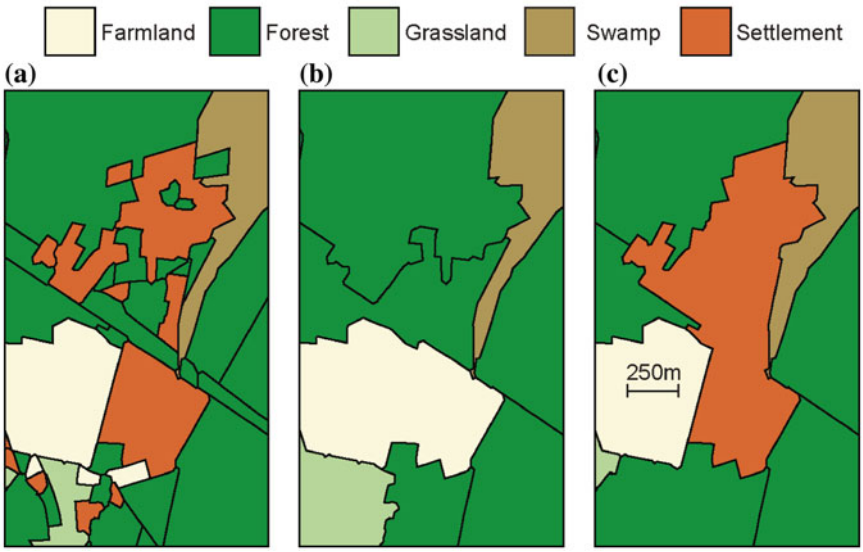

Fig. 4.15 Comparison of two aggregation methods (ATKIS): **a** input dataset DLM50, **b** iterative merging towards DLM250, and **c** optimisation towards DLM250

4.5.1 Create Constraint Dataset Via Optimisation

Figure 4.15 illustrates the generalisation challenge. The sample in Fig. 4.15a was taken from the German topographic database ATKIS DLM50, which contains the same amount of details as a topographic map at the 1:50,000 scale. In order to generalise these data, we need to satisfy size conditions defined for the lower level of detail (LoD) that we aim to achieve. In many countries, such conditions are defined in specifications of topographic databases, for example, in Germany (Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland: ATKIS-Objektartenkatalog: http://www.atkis.de).

Obviously, size conditions can be satisfied by aggregation, that is, by combining the input areas into fewer composite regions. This can be done, for example, by iteratively selecting the least important area in the dataset and merging it with its most compatible neighbour until all areas have sufficient size. Different measures of compatibility and importance have been proposed. Figure 4.15b shows the result of the algorithm when applying size conditions defined for the ATKIS DLM250, which corresponds to scale 1:250,000. Though all areas in the output have sufficient size, the example reveals a shortcoming of the iterative algorithm: As the algorithm only takes the compatibility between adjacent areas into account, large parts of the dataset change their class. In particular, the group of non-adjacent settlement areas is lost. This is because each single settlement area is too small for the target scale and becomes merged with a

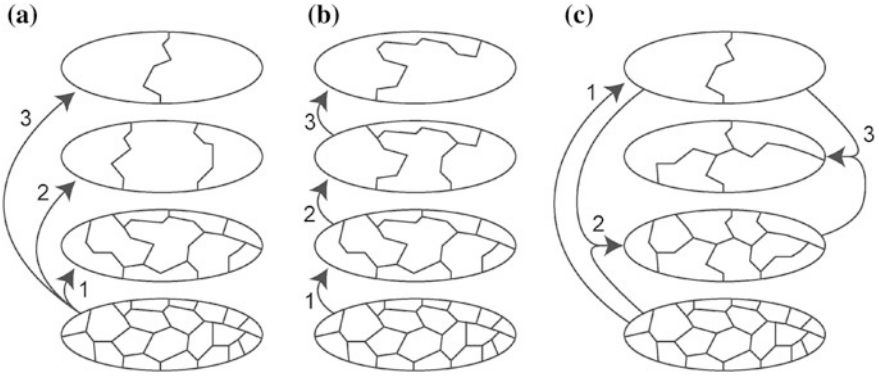

Fig. 4.16 Approaches to create a sequence of LoDs: **a** generalisation from a single source dataset (repeated optimisation), **b** successive generalisation (classic tGAP approach), and **c** intermediate representations fitting in the target representation (constraint tGAP approach)

neighbour of another class, for example, forest. However, together the group of settlements would reach the required size. In order to preserve such groups, we developed an optimisation method for aggregation that minimises the change of land cover classes while ensuring contiguous regions of sufficient size (Haunert and Wolff 2006). The result is shown in Fig. 4.15c: The small settlements are grouped into one large composite region. Generally, constraint- and optimisation-based approaches to map generalisation are considered highly flexible and capable of providing high-quality results.

4.5.2 Use Automatically Generalised (Optimised) Dataset to Create Constraint tGAP

Our basic assumption is that we are given an algorithm for the classical map generalisation problem, that is, for a given input dataset we can produce a dataset at any reduced LoD by appropriately setting the parameters of the algorithm. We can apply our optimisation approach for this task or any other generalisation methods that are available. Figure 4.16 illustrates three different ideas to generate a sequence of LoDs by applying such an algorithm.

In Fig. 4.16a the most detailed dataset is used as input for the algorithm to generate all levels of the sequence. Though each single dataset is well generalised, the obtained sequence of datasets does not conform to the idea of gradual refinement. For example, a line boundary that appears at a smaller LoD may not be present at a higher LoD. An alternative approach is to generate the sequence of LoDs in small steps—each step using the previously generated dataset as input for the generalisation algorithm (Fig. 4.16b). The iterative method that we previously used to set up the tGAP structure follows this approach. Though it leads to a sequence of relatively small changes between two consecutive LoDs, it entails the

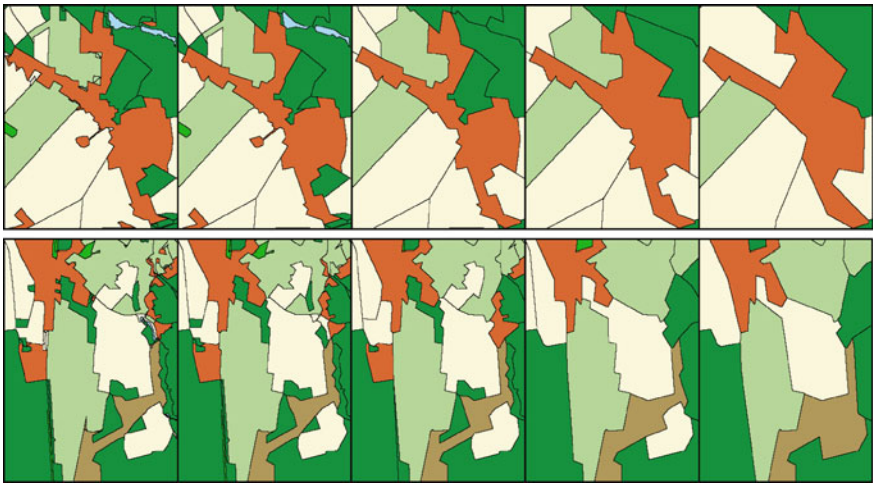

Fig. 4.17 Two examples of optimised constraint tGAP. From *left* to *right* most detailed dataset, a series of intermediate representations, and finally arriving at a dataset generalised by optimisation

risk of getting unsatisfactory results at low LoDs (end result). In particular, this iterative approach does not allow us to optimise global quality measures, for example, to minimise changes of land cover classes between the highest LoD and the lowest LoD. Figure 4.16c shows a third approach, which we propose to overcome the disadvantages of both the other methods: We first create the dataset at the lowest LoD (smallest scale) and then define a sequence of intermediate representations (Fig. 4.16c). Using our optimisation method for the first stage, we can ensure a well-generalised dataset at the lowest LoD. In order to define the intermediate LoDs, we apply a modified version of the iterative algorithm that we earlier applied to set up the tGAP structure. Some results for two examples are visible in Fig. 4.17.

4.6 Case Study III: Corine and ATKIS Data in the Space Scale Cube

Section 4.3 introduced the third dimension to represent the scale of the tGAP structure in the so called Space-Scale Cube (SSC). While Sect. 4.3 only used artificial data sets to illustrate the 3D principle, Sects. 4.4 and 4.5 used real map data to create a constraint tGAP structure. In this section, we will first show an artificial example, followed by some examples with real data from respectively the Corine data set (1:100,000) and the ATKIS DLM data set (1:50,000), illustrating the concept of the SSC. Finally, the results of the 3D SSC will be shown, where gradual-change line simplification is applied. For all data sets first a tGAP structure is created which is then converted into a SSC.

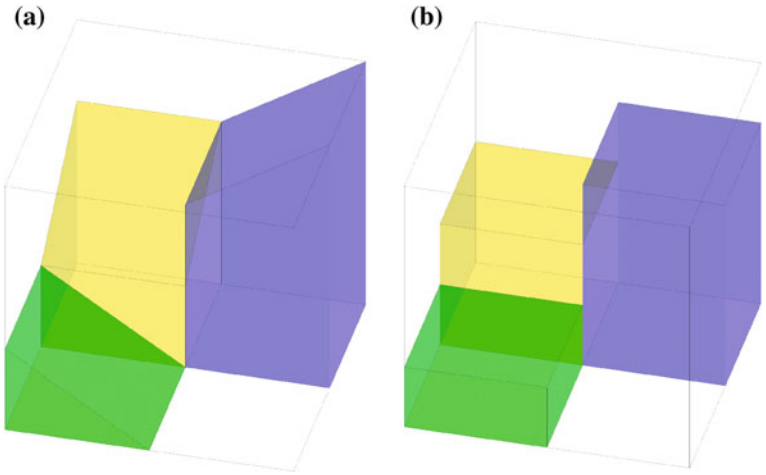

Fig. 4.18 Two tGAP to SSC construction alternatives: **a** prism based approach (transparent/*white* object first taking over space from *green* neighbour, next from *yellow* neighbour and finally from *purple* neighbour and the whole area has become *white*), **b** the gradual transition approach

4.6.1 Artificial Data in SSC

The artificial data example in Fig. 4.18 illustrates the difference of a sudden-change ('discrete') merge operation versus having a smooth ('gradual') merge operation available. The prism based SSC is the result of sudden-change merge operation, resulting in horizontal faces closing the space-scale polyhedrons in the SSC. This leads to sudden changes in the derived 2D maps when the horizontal slicing place passes the horizontal top face of such a prism. Hence, the gradual change for a merge operation was proposed by removing the horizontal top faces of the prism and replacing them by tilted faces (Sect. 4.3.1).

When gradually moving the horizontal slicing plane, this gives a more continuous transformation of the map (a small change in scale leads to a small change of the map). By recursively repeating the same pattern (of Fig. 4.18), a larger artificial data set is created in the form of a checkerboard (Fig. 4.19).

Creating a perspective view by applying a tilted slicing plane we can see how it gradually changes from higher to lower detail in this mixed scale representation. The difference between the perspective views generated for the prism based SSC (sudden-change merge) and the gradual transition SSC is illustrated in Fig. 4.20. Note that the prism based (sudden-change) SSC might look better in this case, because the data are very artificial and the diagonal slicing plane is axis aligned and the rectangular pattern is better maintained. With real world data we expect that the smooth (gradual transition) SSC actually gives better results.

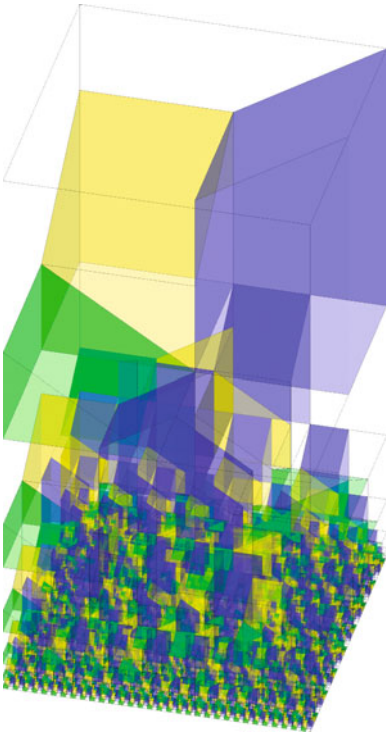

Fig. 4.19 The tGAP to gradual SSC construction for a larger artificial data set (recursive repetition of the same type of transitions)

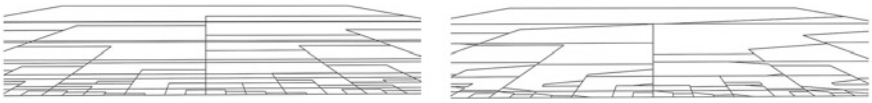

Fig. 4.20 Slicing the two tGAP to SSC construction alternatives (with slicing plane aligned with the *cube*). Note that slicing the more simple prism SSC looks better (no 'diagonals') compared to the more advanced gradual SSC. This is due to the artificial nature of starting with a regular grid

4.6.2 Corine Data in SSC

First, a tGAP was constructed based on polygons from the Corine dataset 2006. The Corine Land Cover (CLC) inventory is a Pan-European land use and land cover map. This dataset is intended to be used at a scale of 1:100,000. Once data are obtained for the tGAP structure, this data can be converted to an explicit 3D structure. Figure 4.21 shows the result for a small subset of the Corine data. For all edges in the tGAP structure vertical faces are constructed. This leads to a SSC that is filled with prism-shaped polyhedrons. From the resulting SSC slices we can see

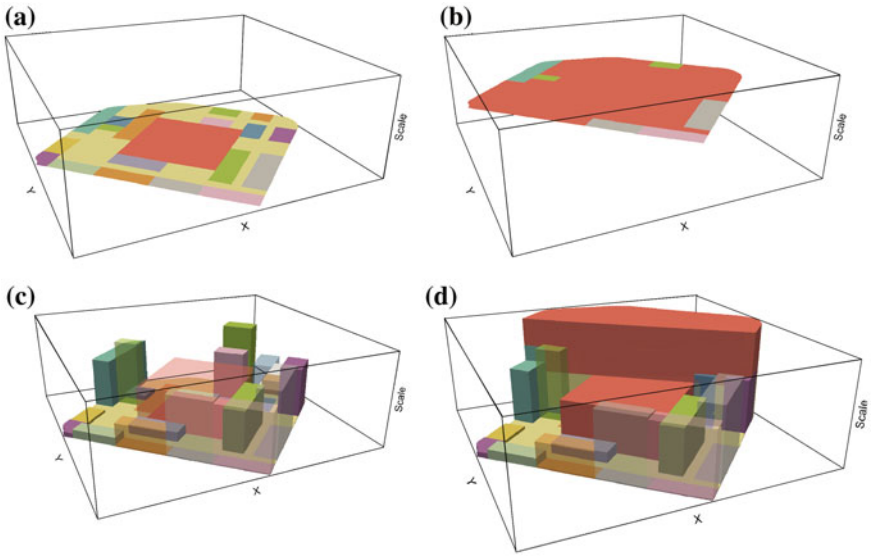

Fig. 4.21 Small Corine subset (30 objects, start scale 1:100,000), based on the SSC representation populated with the prism directly derived from the tGAP: **a** (*top left*) detailed slice low, **b** (*top right*) overview slice high, **c** (*bottom left*) objects in 3D SSC—more important objects are higher and **d** (*bottom right*) objects and tilted plane slice in 3D SCC—the red object are deemed to be the most important and are therefore retained until the end

more detail is available near the bottom (original data) and less (more generalised) data is available near the top of the cube.

4.6.3 ATKIS Data in SSC

Figure 4.22 shows the use of generalised ATKIS data in the SSC. Again the result is a prism based SSC, which was also constructed by extruding all the tGAP edges into vertical faces. Furthermore, Fig. 4.22b illustrates taking a non-horizontal slice, that leads to a map of mixed scale. Such a map is intended to be used in perspective view; Fig. 4.20 gives an impression of how this looks, using artificial data.

4.6.4 Smooth Line Simplification in the SSC

Up till now, the prism-based SSC data was presented (based on only merging neighbours and no line simplification). This section illustrates that the inclusion of line simplification in the gradual-change map generalisation leads to non-vertical faces in the SSC. Figure 4.23 shows two variants of a single space-scale

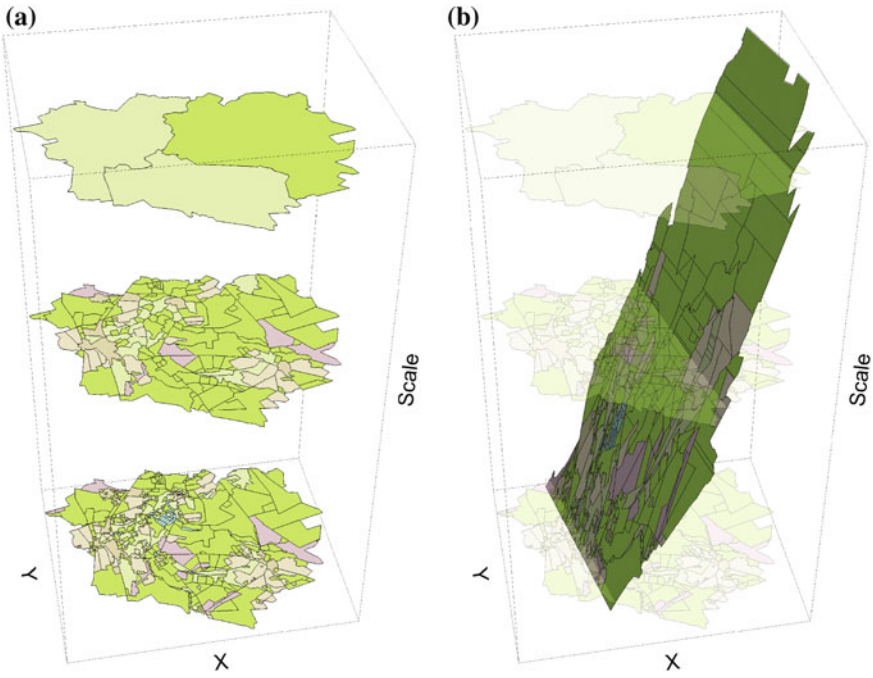

Fig. 4.22 Medium ATKIS subset (1,049 objects, start scale 1:50,000), based on the SSC representation populated with the prism directly derived from the tGAP: **a** detailed slices low, medium and overview slice high, **b** titled plane slice

polyhedron (selected from the complete SSC). The first polyhedron does not show any simplification to its boundaries (Fig. 4.23a). The boundaries are the same for the whole range of the map scales where this polyhedron is used (Fig. 4.23b–d). The second polyhedron shows the space-scale representation for the same object, when the boundary of the object is gradually simplified (Fig. 4.23d). The line simplification was performed on the edges of the tGAP structure, using the approach of Meijers (2011), in which a set of polylines is simplified simultaneously, but in which any changes in the topology are not allowed.

Construction of the first polyhedron, takes as input the polygon and outputs vertical polygons for every segment of the boundary (similar to how the illustrations of real world data in Sects. 4.6.2 and 4.6.3 were made). In the second case, the result of the line simplification algorithm is stored in a BLG tree in the edge table of the tGAP structure. This binary tree structure captures all the knowledge to be able to construct the 3D surface boundaries for every edge in the tGAP structure, making the space-scale cube an explicit 3D representation. If a slice is taken and gradually moved through this 3D model, the result is a smooth changing 2D representation of the map object. This idea is illustrated in Fig. 4.23f–h.

Whether true 3D data storage is to be applied for vario-scale data, depends hugely on the purpose. In an editing environment, a real 3D implementation of 2D

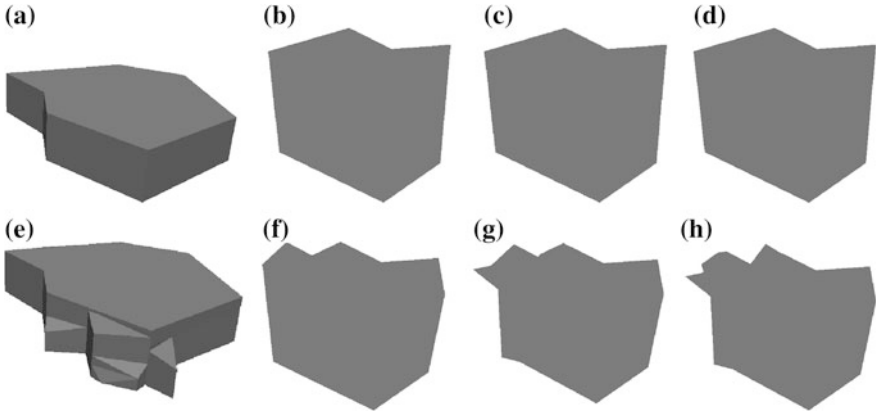

Fig. 4.23 Comparing a space-scale polyhedron (**a**) without and (**e**) with line simplification. The resulting slices (**b**), (**c**) and (**d**) through the polyhedron without line generalisation. All are the same, while the resulting slices (**f**), (**g**) and (**h**) through the polyhedron with line simplification have gradually reducing detail

space and 1D scale data can have benefits for interactive editing sessions. With the right tools available (for example, the editing environment does not allow the creation of intersecting surfaces), this might provide an intuitive way of editing vario-scale data. However, for real time use of the data for end users, where they are using the data as a backdrop map, it may be more efficient if the data is compactly encoded in the original tGAP data structures. 3D surfaces in the Space Scale Cube can be represented efficiently by storing the sequence of generalisation steps that were performed during line simplification in a (set of) BLG tree(s). Such a binary tree can subsequently be compactly serialised into a binary data format, which saves bandwidth during transmission from a client to server. It remains to be seen what is the best approach (explicit 3D cube using a general 3D data structure, or highly specialised 2D data structure, together with separate support for 1D scale). Conceptually however, the explicit 3D model makes it easier to reason about the desired effects of the map generalisation process from a geometric point of view.

4.7 Conclusions

Section 4.2 presented a geometrical non-redundant, variable-scale data structure. The previous versions of the GAP tree had some geometry redundancy, primarily between the polygons at a given scale and/or between the scales. The key to the solution presented in this section was applying a full topological data structure, though this is far more complicated than topological structures designed for the traditional single-scale data sets. The topological GAP tree is very well suited for a

web environment–client requirements are relatively low (no geometric processing of the data at the client side) and progressive refinement of vector data is supported (allowing quick feedback to the user).

Section 4.3 has introduced a truly smooth vario-scale structure for geographic information: a delta in scale leads to a delta in the map continuously down to an infinitesimally 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 on how to obtain data for the smooth tGAP structure by performing generalisation operations in such a way that the output gradually changes the boundaries between the features being generalised.

The smooth tGAP structure delivers 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 smooth vario-scale). The first advantage comes from tight integration of space and scale resulting in 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. Secondly it is relatively easy to implement smooth zoom, and thirdly, it provides a compact (good for storage, transfer and CPU use) and object-oriented encoding (one higher dimensional object for a complete scale range).

In Sects. 4.4 and 4.5, we presented an approach to the integration of datasets of two different scales and generalisation between the scales based on the constrained tGAP structure. Objects of the medium-scale dataset are used for both approaches as constraints in the generalisation process. They restrict the aggregation of the large-scale objects only inside these constraints, resulting in a gradual transformation of the large-scale dataset into the medium-scale dataset. In Sect. 4.4, we used large- and (independent) medium-scale topographic data from the Netherlands. The first stage of the process is object matching between the two datasets. The output of the matching process is used in the second stage, the generalisation that is performed by the constrained tGAP. In Sect. 4.5, the constraints were derived from the large scale data set, by applying an optimisation method. The values that remain crucial for the quality of GAP-tree generalisation are the importance value of the selected feature classes (and importance function) and the compatibility values between two different feature classes (and compatibility function). More research is needed in this area to automatically obtain good generalisation results for real-world data. The constrained approach however results in a significant improvement of the generalisation quality (compared to unconstrained tGAP).

Section 4.6 has shown how a conceptual 3D model is straightforward for deriving smooth representations. The state-of-the-art implementation engineered for 2D + 1D scale separately. However, our initial implementations applied to real data have shown that the approach is indeed feasible.

Although the smooth tGAP structure encoded in the SSC as presented in Sect. 4.3 is a breakthrough vario-scale data structure supporting smooth zoom, there is still a myriad of open research questions: Efficient engineering implementation issues, cf. van Oosterom et al. (2002), Meijers et al. (2009), formalise the structure and prove mathematical proofs to be based on Meijers and van Oosterom (2011) and Thompson and van Oosterom (2012), tune the creation process (an option is to use the constrained tGAP; Haunert et al. 2009), test with larger real world data sets and appropriate graphical user interfaces, investigate the effects of the Collapse/Split operator, improve the cartographic generalisation quality (include more generalisation operators such as displacement, typification, and symbolisation; Cecconi and Galanda 2002), parallel generalisation (and not tGAP-style 'one by one sequencing' of the generalisation operations), cartographic styling of smooth tGAP selections (which are mainly a DLM; Stoter et al. 2010), dynamic map labeling Been et al. (2010), support for non-linear primitives (e.g. circular arcs, cylinder/sphere patches, non-uniform rational B-splines, NURBS; Pu and Zlatanova 2006), and make the structure dynamic 3D objects, scale and time lead to 5D hypercube; van Oosterom and Stoter 2010.

Acknowledgments 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 which is partly funded by the Ministry of Economic Affairs.

Many thanks to all students and colleague researchers involved in the development of the vario-scale tGAP concepts: Vincent Schenkelaars, Judith van Putten, Tinghua Ai, Maarten Vermeij, Arjen Hofman, Arta Dilo, Marian de Vries and Jan-Henrik Haunert. Thanks to Wilko Quak, Edward Verbree, and Rod Thompson for a critical review of parts of draft version of this text. Finally, authors would like to thank the editors of this book for the initiative and for, together with the anonymous reviewers, providing many constructive suggestions on the earlier versions of this chapter. Special thanks to William Mackaness for his meticulous English proof reading and spotting unclear sections. All comments were of great help, but all (remaining) errors are the sole fault of the authors.

References

Ai T, van Oosterom P (2002) GAP-tree extensions based on Skeletons. Paper presented at the advances in spatial data handling, 10th international symposium on spatial data handling, Ottawa, Canada, 9–12 July 2002

Ballard DH (1981) Strip trees: a hierarchical representation for curves. Commun Assoc Comput Mach 14(5):310–321. doi:10.1145/358645.358661

Been K, Nöllenburg M, Poon S-H, Wolff A (2010) Optimizing active ranges for consistent dynamic map labeling. Comput Geom 43(3):312–328

Bertolotto M, Egenhofer MJ (2001) Progressive transmission of vector map data over the World Wide Web. Geoinformatica 5(4):345–373. doi:10.1023/a:1012745819426

Bregt A, Bulens J (1996) Application oriented generalization of area objects. M Methods for the generalization of geo-databases. Netherlands Geodetic Commission, Delft, The Netherlands, pp 57–64

Buttenfield B (2002) Transmitting vector geospatial data across the internet. In: Egenhofer M, Mark D (eds) Geographic information science, vol 2478. Lecture notes in computer science. Springer Berlin Heidelberg, pp 51–64. doi:10.1007/3-540-45799-2_4

Cecconi A, Galanda M (2002) Adaptive zooming in web cartography. Comput Graph Forum 21(4):787–799. doi:10.1111/1467-8659.00636

DMA USDMA (1986) Defense Mapping Agency product specifications for digital feature analysis data (DFAD) Level 1-C and Level 3-C (DFAD): level 1 and level 2. DMA Aerospace Center, St. Louis, Mo

Douglas DH, Peucker TK (1973) Algorithms for the reduction of the number of points required to represent a line or its caricature. Cartographica: Int J Geograph Inf Geovisual 10(2):112–122. doi:10.3138/FM57-6770-U75U-7727

Gunther O (1988) Efficient structures for geometric data management, vol 337. Lecture notes in computer science. Springer, Berlin

Guttman A (1984) R-trees: a dynamic index structure for spatial searching. SIGMOD Rec 14(2):47–57. doi:10.1145/971697.602266

Hägerstrand T (1970) What about people in regional science? Pap Reg Sci 24(1):6–21

Hampe M, Sester M, Harrie L (2004) Multiple representation databases to support visualization on mobile devices. In: Proceedings of the 20th ISPRS congress, vol 35 of international archives of photogrammetry, remote sensing and spatial information sciences, Istanbul, Turkey, vol 35(6), pp 135–140

Harrie L, Sarjakoski T, Lehto L (2002) A variable-scale map for small-display cartography. Int Arch Photogrammetry Remote Sens Spat Inf Sci 34(4):237–242

Haunert J-H, Wolff A (2006) Generalization of land cover maps by mixed integer programming. Paper presented at the proceedings of the 14th annual ACM international symposium on advances in geographic information systems, Arlington, Virginia, USA

Haunert J-H, Dilo A, van Oosterom P (2009) Constrained set-up of the tGAP structure for progressive vector data transfer. Comput Geosci 35(11):2191–2203

Hildebrandt J, Owen M, Hollamby R (2000) CLUSTER RAPTOR: dynamic geospatial imagery visualisation using backend repositories. In: Proceedings of the 5th international command and control research and technology symposium (ICCRTS) 2000

Hofman A (2008) Developing a vario-scale IMGeo using the constrained tGAP structure. MSc thesis geomatics, Technical Delft University

Jones CB, Abraham IM (1987) Line generalisation in a global cartographic database. Cartographica: Int J Geograph Inf Geovisual 24(3):32–45. doi:10.3138/0666-1648-61L4-3164

Jones C, Abdelmoty A, Lonergan M, van der Poorten P, Zhou S (2000) Multi-scale spatial database design for online generalisation. In: 9th international symposium on spatial data handling, pp 34–44

Kreveld M (2001) Smooth generalization for continuous zooming. In: Proceedings of 20th international cartographic conference, Beijing (China), pp 2180–2185

Lazaridis I, Mehrotra S (2001) Progressive approximate aggregate queries with a multi-resolution tree structure. Paper presented at the proceedings of the 2001 ACM SIGMOD international conference on management of data, Santa Barbara, California, USA

Meijers M (2011) Simultaneous and topologically safe line simplification for a variable-scale planar partition. In: Geertman S, Reinhardt W, Toppen F (eds) Advancing geoinformation science for a changing world. Lecture notes in geoinformation and cartography. Springer Berlin Heidelberg, pp 337–358. doi:10.1007/978-3-642-19789-5_17

Meijers M, van Oosterom P (2011) The space-scale cube: an integrated model for 2D polygonal areas and scale. Paper presented at the 28th urban data management symposium, volume 38 of international archives of photogrammetry, remote sensing and spatial information sciences

Meijers M, van Oosterom P, Quak W (2009) A storage and transfer efficient data structure for variable scale vector data. In: Sester M, Bernard L, Paelke V (eds) Advances in GIscience.

Lecture notes in geoinformation and cartography. Springer Berlin Heidelberg, pp 345–367. doi:10.1007/978-3-642-00318-9_18

Noguera JM, Segura RJ, Ogáyar CJ, Joan-Arinyo R (2011) Navigating large terrains using commodity mobile devices. Comput Geosci 37(9):1218–1233. doi:10.1016/j.cageo.2010.08.007

Pu S, Zlatanova S (2006) Integration of GIS and CAD at DBMS level. In: Proceedings of UDMS 2006, pp 9.61–69.71

Rosenbaum R, Schumann H (2004) Remote raster image browsing based on fast content reduction for mobile environments. Paper presented at the proceedings of the seventh Eurographics conference on multimedia, Nanjing, China

Samet H (1984) The quadtree and related hierarchical data structures. ACM Comput Surv Arch 16(2):187–260. doi:10.1145/356924.356930

Sester M, Brenner C (2005) Continuous generalization for visualization on small mobile devices. In: Developments in spatial data handling. Springer, Berlin Heidelberg, pp 355–368. doi:10.1007/3-540-26772-7_27

Stoter J, Meijers M, van Oosterom P, Grünreich D, Kraak M-J (2010) Applying DLM and DCM concepts in a multi-scale data environment. In: Buttenfield BP, Brewer CA, Clarke KC, Finn MP, Usery EL (eds) Proceedings of GDI 2010: symposium on generalization and data integration 2010, pp 1–7

Thompson RM, van Oosterom P (2012) Modelling and validation of 3D cadastral objects. Urban and regional data management—UDMS annual 2011, pp 7–23

van Oosterom P (1986, 1989) A reactive data structure for geographic information systems. In: AutoCarto 9, Baltimore, Maryland, pp 665–674

van Oosterom P (1990) Reactive data structures for geographic information systems. PhD Theses, Department of Computer Science, Leiden University, The Netherlands

van Oosterom P (1992) A storage structure for a multi-scale database: the reactive-tree. Comput Environ Urban Syst 16(3):239–247. doi:10.1016/0198-9715(92)90036-Q

van Oosterom P (1993) The GAP-tree, an approach to "On-the-Fly" map generalization of an area partitioning. GIS and generalization, methodology and practice. Taylor & Francis, London

van Oosterom P (1994) Reactive data structures for geographic information systems. Oxford University Press, Inc.

van Oosterom P (2005) Variable-scale topological data structures suitable for progressive data transfer: The GAP-face tree and GAP-edge forest. Cartogr Geogr Inf Sci 32(4):331–346. doi:10.1559/152304005775194782

van Oosterom P, Meijers M (2011a) Method and system for generating maps in an n-dimensional space. Dutch patent application 2006630, filed 19 April 2011, published October 2012

van Oosterom P, Meijers M (2011b) Towards a true vario-scale structure supporting smooth-zoom. In: Proceedings of 14th ICA/ISPRS workshop on generalisation and multiple representation 2011, pp 1–19

van Oosterom P, Schenkelaars V (1995) The development of an interactive multi-scale GIS. Int J Geogr Inf Syst 9(5):489–507. doi:10.1080/02693799508902052

van Oosterom P, Stoter J (2010) 5D data modelling: full integration of 2D/3D space, time and scale dimensions. Paper presented at the proceedings of the 6th international conference on geographic information science 2010, Zurich, Switzerland

van Oosterom P, Stoter J, Quak W, Zlatanova S (2002) The balance between geometry and topology. In: Richardson D, van Oosterom P (eds) 10th international symposium on spatial data handling. Springer, Berlin, pp 121–135. doi:10.1007/978-3-642-56094-1_16

van Putten J, van Oosterom P (1998) New results with generalized area partitionings. In: Proceedings 8th international symposium on spatial data handling 1998, pp 485–495

van Smaalen JWM (1996) A hierarchic rule model for geographic information abstraction. In: Proceedings, SDH'96, Delft, The Netherlands, p 4b.31

van Smaalen JWM (2003) Automated aggregation of geographic objects—a new approach to the conceptual generalisation of geographic databases. PhD Theses, Wageningen University, The Netherlands

Vermeij M, van Oosterom P, Quak W, Tijssen (2003) T Storing and using scale-less topological data efficiently in a client-server DBMS environment. In: Proceedings of the 7th international conference on geocomputation, pp 1–11

Zhou X, Prasher S, Sun S, Xu K (2004) Multiresolution spatial databases: making web based spatial applications faster. In: Proceedings of the 6th Asia-Pacific web conference, APWeb. Springer, pp 36-47