A+BE Architecture and the Built environment

#18 2017

Design and development of a system for vario-scale maps

Radan Šuba

Design and development of a system for vario-scale maps

Radan Šuba
Delft University of Technology, Faculty of Architecture and the Built Environment,
OTB – Research for the Built Environment,
Section GIS technology

Architecture
and the
Built environment

abe.tudelft.nl

Design: Sirene Ontwerpers, Rotterdam

ISBN 978-94-92516-94-7
ISSN 2212-3202

Acknowledgements

I have remembered the moment approximately 66 months ago. I was working on my master thesis at that time. I had a consultation with my former supervisor Karel Janečka. At some point he said: "By the way, there is an open position for PhD in the Netherlands. Think about this. If I would be young, I would go for it." Next day I was sending my application and "boom" here we are, at the end of the long journey.

I was not always alone on the way. I was accompanied by nice people. Here I would like to take this opportunity to express my sincere gratitude to those people who provide help and support to me, to my research and make all this possible.

First of all, I would like to thank my family. Great part goes to my parents, my sisters Lucka and Šárka and their families.

Thanks also to all people I have had the pleasure to meet. Rūta, Xin and Ufa shared our open space office and we had many refreshing discussions. Ana, Elise, Igor, Jaap and others spend many coffee breaks with me. Thanks guys. Thanks to my Iranian friends Dena, Pirouz, Sina and Aida for their big understanding and their healthy attitude full of irony. I appreciated that. I was also lucky to share accommodation with Mladen and latter with Konstantin; two easygoing and tidy guys, which made life much easier. Time to time it was nice to speak my own language, and thanks to other Czechs living in Delft Anežka, Jirka and Zuzka, I could. Thanks to Blue Falcons, great sport club full of enthusiastic people. It was nice to play together with Lauris, Christian, Marcel and others. It was my pleasure to be part of such a great floorball team even though we have lost majority of our games.

Furthermore, I feel lucky that so many people find the way to the Netherlands to visit me. Every visit was special and I enjoyed every one of them. It helped keep my spirit up. Péťa, Lucka and rest of my family who were the first. My cousin Martin with Dana, Vladan and Radim who were the most enthusiastic. There were many more; Bětka and Caesar, Domka and Pítr, Zelí, Skip, Ondra, and Eliška and Holi who were amused by Dutch phrase 'extra belegen'. Also, thank to Kamilka who came to save me and kept my sanity high.

I was also fortunate for meeting so many nice people in GIS field. Thanks to Ravi for many and various things from nice times lunching together, through helping me out with a small python hurdle to becoming a friend. All other GIS geeks Ken, Filip, Liu Liu, Oscar, Zhiyong for hearing me out and answering my GIS related questions. A special word of gratitude goes to Elfriede for her help with English checks and all kinds of administrative issues; Marian for pleasant company in our office where we sit side by side – in spite of the fact that it was the most of the time in silence. Theo, Tjeu, Wilko, and Edward thank you for your kind support, nice conversations, and inspiring discussions.

Special thanks goes to Rod Thompson who read the thesis from cover to cover, gave so many useful comments and corrected my English. Without him the thesis would be much less readable.

Finally, my biggest thanks go to Martijn Meijers and Peter van Oosterom. I will always remember some of our meetings when the constructive planning was slowly transformed to exciting brainstorming – when we ended up with numerous sketches and drawings on many pieces of paper capturing our ideas and visions. Thanks to Peter who gave me the opportunity to be a part of R&D on such a fascinating concept; for his leadership which I could grow under; for unlimited support; for his always positive spirit which motivates me a lot and helped me keep running for such a long time on the same topic. Thanks to Martijn for his limitless help; for his guidance on my path; for tutoring; for sharing with me all the stuff which GIS world offers; for answering all my questions even the most stupid as it could seem.

Thank you all
Radan

Contents

1 Why do we need vario-scale maps?

Maps have always played a significant role in history. They have helped sailors on their Pacific journeys, they have provided information to generals for planning invasions, and they have been used for more pragmatic reasons, such as tax collecting. Nowadays, maps are even more important because they are part of many fields and easily accessible in every smartphone; however, their form differs. While maps on paper still exist, most of them are transferred via the Internet, where users could in principle use the full potential of interactive computer environment, *e. g.* zooming. Nevertheless, with respect to their creation, retrieval and visualization, maps have stayed more or less the same as in the past. Therefore, this thesis focuses on map creation processed more suitable for interactive computer environment.

In the next section (Section 1.1), the main issues of the current solution are introduced. Section 1.2 suggests an advantages of the researched system. Main research questions covered in this thesis follow in Section 1.3. Furthermore it highlights the research scope in Section 1.4, the research methodology in Section 1.5, and gives an overview of the thesis structure in Section 1.6.

§ 1.1 Paper maps in digital enviroment

One of the main limitations of the classical approach to map making is the concept of scale, similar to the concept of Level of Detail (LOD), where the scale of a map is defined as the ratio of a distance on the map to the corresponding distance in real world. When moving to a computer environment, the old map scale concept is maintained with redundant data and content overlaps between scales (with potential inconsistency).

Other consequences of the classical 'paper' approach is the redundancy of data. Every time, when a user zooms in or out, a new predefined separate map is retrieved. With many separate fixed levels, a lot of information is the same, but still needs to be included, *i. e.* the identical map features are depicted in the map again and their geometries are stored multiple times in the database. This leads to duplication, more data transfer and slower response, because the same data must be sent from the server to the user's device repeatedly.

Maintenance and updates of the data are also an issue. The same features at different levels lack any connections most of the time; for example the same lake at two different scales is represented by two different objects. Then, sometimes even the name must be independently updated, the multiple representations (scales) must be visited, checked and modified. Therefore, this lack of links between data at different scales results in more complex analysis, search or processing.

A final issue is related to user navigation. The interactive Internet environment offers smooth animated changes of the content, or gradual transition between feature representations; for example when a user zooms in towards the most detailed level, the geometry of the building gradually appears. These dynamic changes are the result of the continuous map approach. However, continuous generalization is not applied to maps presented on the Internet. There are already map user interfaces providing that feeling by simulating a smooth zoom, *e. g.* Google Maps[1], Mapbox[2] and Microsoft Bing Maps[3]. However, this is just an illusion. Their solution is still based on a large number of redundant and fixed map scale representations.

Together, these issues result in a labour intensive and expensive process in the manner of handing geo-informations at range of scales, which in practice results in different and sometimes conflicting versions of the same data appearing in different places, so-called inconsistencies. Therefore, a new approach for a digital environment is needed.

§ 1.2 Vario-scale

We use a concept named 'variable-scale' (vario-scale for short) proposed by van Oosterom et al. (2014), where the data once stored in a vario-scale data structure can be used for generating all desired (needed) scales in a smooth digital way. The main idea of our alternative approach is based on the specific vario-scale data structure called tGAP (topological Generalized Area Partition) (van Oosterom, 2005; van Oosterom and Meijers, 2011b). This structure actually represents the results of map generalization actions; features are generalized in small steps, progressively leading to a simpler and simpler map.

We assume that map objects can be well generalized with optimized algorithms with appropriate parameters for use at any map scale. Therefore, we make the whole map repeatedly simpler and simpler, where the least important feature in the map is simplified based on a global criterion. These generalization steps are stored in the vario-scale data structure, which captures these incremental changes with minimal redundancy. Both the detailed objects at the largest scale and the objects generated during the generalization process are represented in a set of database tables.

Redundant storage is avoided as much as possible; instead of the explicit geometrical representation for every face as polygon, it stores only the shared boundaries between neighbouring faces (edges in the topological sense) in the specific structure. Once the automated generalization process is finished, a valid range of map scales is defined for every topological primitive (node, edge or face) in the structure. These primitives including scale ranges can then be used to construct maps at arbitrary map scale.

All objects are stored as 2D objects in a topological data structure (tGAP) extended with scale ranges. Up to now, the data have been stored and visualized as 2D maps, but this did not result in true smooth zooming operation (sudden small changes in map

1 `www.maps.google.com`

2 `www.mapbox.com/gallery/`

3 `www.bing.com/maps`

content are visible during use as result of specific generalization action). Therefore, a 3D structure called Space-Scale Cube (SSC) (Meijers and van Oosterom, 2011) where objects are represented as 3D volumetric data (2D geometry + 1D scale) has been proposed. In this structure, these (small) steps are represented by gradual transitions. However, this theoretical concept has not yet been implemented prior to the start of this PhD research.

Vario-scale approach is ongoing research proposed in multiple publications. These studies have provided theoretical and technical background for the vario-scale concept in general. However, there are still a significant number of unproven suggestions for improvement. Therefore, further design and development with respect to vario-scale cartographic map content generation and use is the main goal of this thesis.

§ 1.3 Research questions

The vario-scale research has been carried out for some years already (within the TU Delft, GIS technology group) with some principles suggested and proven and many more not yet proven. The main question at this research is the following:

MAIN: *How to design and develop a system for vario-scale maps?*

This research question is valid for whole vario-scale research theme and it will be main question of this thesis. From here more specific sub-questions are defined connecting to previous research (Meijers, 2011). Therefore, the sub-questions we want to address are following (together with corresponding chapter within brackets):

SUB I: *How much is the current map generalization process automated?* (Chapter 2)

This will give us solid starting position for our research. It also shows state-of-the-art solutions of map generalization. Since the vario-scale concept is known for some time already it investigates if such research about automated generalization still has meaning. This knowledge is needed to drive the generalization process of which the result is stored in a vario-scale structure.

To provide the foundation for this thesis and to understand the whole vario-scale concept together with the reasoning behind, we need to answer:

SUB II: *What is the state of the art vario-scale map?* (Chapter 3)

The previous investigated structures only explicitly support area features and the content/cartographic quality was not optimal. Other features (line and point) are not yet included explicitly in the structure. However, these types of objects are important for maps. Some line features are already implicitly produced during the creation of the tGAP structure. The collapse of an area feature is a good example, *i. e.* the area of road is converted to a line. It is convenient to store information about the collapsed feature as well. Therefore, we are looking for an answer of the question:

SUB III: *How can we produce improved vario-scale data content?* (Chapter 4)

Often, the efficiency of early solution is bounded by the size of datasets. Since geographical data are typically massive, they do not fit in the main memory of computer. This is a challenging process in general, but especially true in cases, in which the relationships between (nearby) features in the map must be considered. Therefore, we pose the question:

SUB IV: *How can we create vario-scale data content not fitting in main memory?* (Chapter 5)

Our approach in theory could generate map content supporting smooth gradual changes in user map interaction when treating scale as third dimension and using the 3D geometries where smooth zooming in or out is thus equivalent to the vertical movement of a horizontal slice plane (downwards or upwards). However, there are no practical experiences or implementations prior to this PhD research yet for either creating nor using this 3D structure. Therefore, the last questions are:

SUB V: *How can we generate, use and validate vario-scale data to achieve a smoother impression by the user?* (Chapter 6)

The answers will give us a tool to verify theoretical concepts about usage of the 3D geometries for smooth zooming more practically. Further, to test whether vario-scale map is perceived better than multi-scale map we can raise the question:

SUB VI: *How do the users perform with vario-scale map compare to multi-scale map?* (Chapter 6)

To reach the better vario-scale maps these research questions should be answered. Furthermore, to make the thesis more readable, clear and structured, every specific question will be covered in one chapter (except the last two).

§ 1.4 Scope

The following list of topics, conditions or domains define the scope of the research:

- The whole research is carried out with vector data only (no raster nor point cloud data). 2D input data are considered.
- Detailed description is from 2D GIS development perspective mainly.
- Examples and test cases with real data are used to demonstrate and evaluate proposed vario-scale solutions.
- Development prioritizes Digital Landscape Model (DLM) containing spacial representation of objects from real world; in contrast to a model targeting production of maps (i. e. geometric representation of objects is adapted to the styling of the map), the so-called Digital Cartographic Models (DCM)(Stoter et al., 2010).

It is also important to say what is not in the scope of this thesis. The following is a list of topics that are in support of this research, but beyond its scope:

- Cartographic design of high quality paper maps (as the emphasis here is more on interactive map use).
- Cartographic map design for various representations between scales, as with increasing number of scales the map levels may present differences in terms of representation.
- The generic output of our work is 2D maps. This implies that 2D resulting maps are our main focus, 3D geometries of higher dimensions are used to support better 2D output.
- Integration with other data. Cases when a base map (in vario-scale) is combined with, for instance, a thematic foreground map are left out.
- Improving data transfer (including data compression) from the server to the end user.
- Dynamic structure. The current tGAP structure is a static one. However, changes of data over time and recomputing the structure on-the fly is out of scope.
- Text labels or symbols of Points of Interest (POI) are also an important part of map content, but these will not be considered in the research.
- Integration of the vario-scale approach within the framework of Open Geospatial Consortium (OGC) standardisation.

§ 1.5 Methodology

This thesis is part of the bigger ongoing line of generalization research towards supporting vario-scale maps. An example of a vario-scale data structure is the tGAP data structure proposed in (van Oosterom, 2005) and since then extensively investigated and further developed.

Various aspects were covered in the multiple earlier publications, including PhD thesis by Meijers (2011) and even registered as a patent (van Oosterom and Meijers, 2011a). Another important characteristic of the research from the early beginning till now has been international cooperation, *e. g.* , with Wuhan University, the University of Hanover or Universiti Teknologi Malaysia (also known as UTM) which leads to multiple publication such as (Ai and van Oosterom, 2002; Dilo et al., 2009; Huang et al., 2016) and provide important reflection from different perspective. Moreover, our research team is an active member of the Commission of Generalisation and Multiple Representation of International Cartographic Association (ICA) and participate in annual Generalisation and Multiple Representation workshops. These activities give the possibility to share our experience and receive important feedback from the experts in the field.

Last four years were funded by the Technology Foundation STW in the project called 'Vario-scale geoinformation' (project code 11185) and the vario-scale research objectives fitted within the definition of this project. The aim of the Technology Foundation STW is to realise knowledge transfer between technical sciences and users. To this end, STW brings researchers and (potential) users together. The instrument par excellence in this respect is the user committee which is also the primary valorization instrument. The user committee meets twice a year and gives feedback (reflection on the research results and suggestion for further direction of the research). The members of the committee are experts in the GIS field: Dutch Kadaster, Rijkswaterstaat (RWS), the municipalities of Amsterdam, Rotterdam and The Hague and Geo-ICT industry: Bentley System Europe B.V., ESRI Nederland B.V., 1Spatial Group Ltd. and Oracle. The committee meetings provide work frame for the project and it helps plan, design and validate the project in half a year time intervals.

In addition to this, we proceed in the iterative way similar to software development process described in (Tutorials Point, 2015); The process of development runs in cyclic manner repeating every development step after every cycle of the process, in so-called iterations. In our case it can be summarized as following: First the theory is developed, make a solution for small test data set, test a solution against real world data, validate, adjust theory and draw conclusions. Then, every following iteration, solution is improved, more features and modules are designed, coded, tested and added to the prototype. Every iteration produces a solution which is complete in itself and has more features and capabilities than that of the previous one. Our prototype is open source and it is shared via online repository. It can be found on following website:

`varioscale.bk.tudelft.nl`

The final result of the process is vario-scale map in tGAP structure. Therefore, it is also important to mention how the quality of the result is measured. Most of the time we define and use quality indicators which can be retrieved automatically from the process such as feature size, the number of objects per features class and area distributions etc. More subjective measures are also involved in the process; visual inspection of map quality, visual comparison with results of previous iteration, and usability testing to limited extend.

§ 1.6 Outline of the thesis

This thesis is organized in the structure which is depicted in the Figure 1.1. It reflects the specific content of individual chapters and relationships among them. They should be perceived as independent contributions and also as closely related (overlapping) components of the unique solution called vario-scale maps. We acknowledge the developing approach where created tools for specific aspect of the problem can be reused later in the process again.

Since map generalization is a rapidly developing field, Chapter 2 gives an overview of map generalization with a focus on automated and continuous map generalization. It introduces related work where maps without fixed target scales with smooth transition are produced. Additionally, it presents state-of-the-art technology for generalization on a national level, because National Mapping Agencies have made enormous improvements in automated generalization lately. These agencies provide solutions characterized by the size of the dataset (millions of records), but also by complexity of their solution (many map features together).

This will provide the base for Chapter 3 where the detailed description of our solution for automated continuous generalization is introduced, resulting in vario-scale maps. Chapter 3 covers the whole vario-scale concept including several (untested) suggestions for better vario-scale maps, recent developments and the process design of the

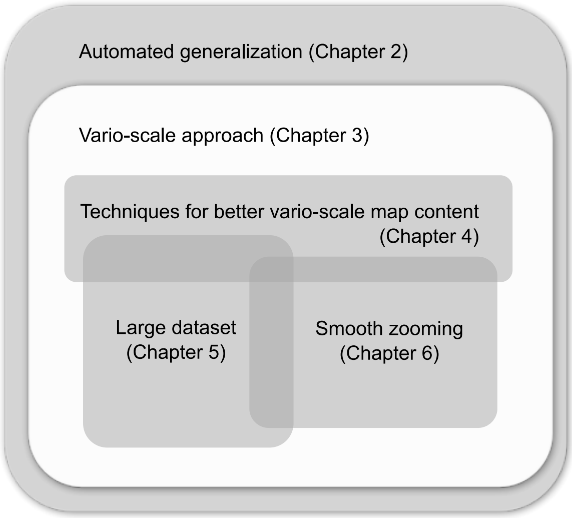

Automated generalization (Chapter 2)
Vario-scale approach (Chapter 3)
Techniques for better vario-scale map content (Chapter 4)
Large dataset (Chapter 5)
Smooth zooming (Chapter 6)

FIGURE 1.1 Outline of the thesis

generalization. It will specify vario-scale framework for the whole thesis where more specific issues will be addressed.

Therefore, one of the issue is to further design and develop the current techniques for better vario-scale map content, develop/find algorithms for the various suggested improvements, and assess the effect when applied to real data. Based on this deeper insight and enriched experiences; again provide/develop suggestion for further improvements. For example, the earlier solution did not consider the different data density of the map *i. e.* very dense urban area compare to sparsely populated rural region, or the fact that the vario-scale generalization process runs throughout whole scale ranges. Therefore, these limitations will be explored in Chapter 4 with specific attention to road map network. We will present developed process from large scale, where roads are represented as area objects, to mid and small scales, where roads are represented as line objects. This generalization of the road network throughout whole scale range is only one of the issues how to design, implement and enrich tools for better vario-scale content covered in Chapter 4.

Another aspect is handing real world dataset containing millions of records. If we want to create vario-scale structure of real world, we should know how to deal with such sizes. Therefore, special attention is paid to this in Chapter 5, where we choose a strategy of splitting the dataset in smaller parts. After that every part has to be computed separately and in the end these have to be merged together. This is repeated at several overlapping levels to avoid hard/remaining boundaries. Chapter 5 describes how to orchestrate such a process.

Later in the process of creating vario-scale maps, when users are involved, zooming in and out of a digital map, it is often necessary to modify the shape of the map. If this is done abruptly, it leads to big changes in geometry, perceived by the user as a 'jump' on the screen. Therefore, Chapter 6 to present smooth zooming operations to the user. This is based on the assumption that every 2D feature in the map is represented in 3D, where the 2D coordinates are the original representation, and the third dimension represents the scale value.

Chapter 7, finally, concludes all aspects, provides critical overview and reflection based on previous chapters, which can lead to further development of vario-scale concept with its promising potential, reduced redundancy, more functionality and better user perception than current classical fixed-scale (multi-scale) maps on the Internet.

Table 1.1 shows an overview of the publications on which the chapters in this thesis are based.

TABLE 1.1 Publications and their relation with the chapters of this thesis.

Ch.	Publication	No.
3	van Oosterom, P., Meijers, M., Stoter, J., and Šuba, R. (2014). *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, volume 2014 of *Lecture Notes in Geoinformation and Cartography*, chapter Data Structures for Continuous Generalisation: tGAP and SSC, pages 83–118. Springer International Publishing.	1.
4	Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014b). Continuous Road Network Generalisation. In *Proceedings of the 17th ICA Workshop on Generalisation and Multiple Representation, Vienna, Austria, September 23, 2014*, pages 1–12.	2.
4	Šuba, R., Meijers, M., and van Oosterom, P. (2015). Large scale road network generalization for vario-scale map. In *Proceedings of the 18th ICA Workshop on Generalisation and Multiple Representation, Rio de Janeiro, Brazil, 21 August, 2015*, pages 1–10.	3.
4	Šuba, R., Meijers, M., and Oosterom, P. v. (2016b). Continuous road network generalization throughout all scales. *ISPRS International Journal of Geo-Information*, 5(8):145.	4.
5	Meijers, M., Šuba, R., and van Oosterom, P. (2015). Parallel creation of vario-scale data structures for large datasets. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-4/W7:1–9.	5.
6	Šuba, R., Meijers, M., and van Oosterom, P. (2013). 2D vario-scale representations based on real 3D structure. In *Proceedings of the 16th ICA Generalization Workshop*, pages 1–11. The paper for ICA worshop 2013/Dresden.	6.
6	Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014a). An area merge operation for smooth zooming. In Huerta, J., Schade, S., and Granell, C., editors, *Connecting a Digital Europe Through Location and Place*, Springer Lecture Notes in Geoinformation and Cartography, pages 275–293. Springer International Publishing. ISBN: 978-3-319-03611-3.	7.
6	Huang, L., Meijers, M., Šuba, R., and van Oosterom, P. (2016). Engineering web maps with gradual content zoom based on streaming vector data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:274 – 293.	8.
6	Šuba, R., Driel, M., Meijers, M., van Oosterom, P., and Eisemann, E. (2016a). Usability test plan for truly vario-scale maps. In *Proceedings of the 19th ICA Workshop on Generalisation and Multiple Representation, Helsinki, Finland*.	9.

2 State of the art in automated map generalization

Automated map generalization is a difficult, complex and computational very intensive problem. The aim of this chapter is to study existing solutions and state of the art. It also provides context and motivation for why we tackle this problem by applying vario-scale approach. In Section 2.1, the paradigm shift in map generalization in a digital environment is studied. We investigate if requirements in the map making process have changed with the transformation from paper to digital environment and if so what are the consequences. Then Section 2.2 investigates how National Mapping Agencies are dealing with automated generalization process in general and what are their recent developments. In Section 2.3, the focus is on the issue of continuous map generalization, which is becoming more researched as an alternative to the map generalization for discrete predefined scales. Section 2.4 demonstrates another problem of digital map environment where the number of map scales available is not sufficient for user interactions. Final remarks are covered in 2.5.

§ 2.1 Dilemma of the generalization in a digital world

The map making process has changed significantly in last decades. In the past, where the map was distributed on the paper cartographers played unique role in the process. They decided based on their experiences how the final map should look. The main focus was on cartographic quality of the resulting maps.

This situation has changed drastically with the change of environment, where the maps are distributed– from paper map to on-line services on the Internet.

Nowadays, the Internet and computers in general offer an environment where data can be generated, stored and distributed in real time. This creates other demands on map making process besides cartographic quality. One, massive amount of data must be effectively stored and managed. Two, geographical data must be transferred fast, even with limited bandwidth. Three, the response time should be minimal. Four, intuitive, easy navigation should be standard, even at a device with small display such as a mobile phone. Four, provided data should be up-to-date. Moreover, there are also other demands such as minimizing the production costs and easy maintenance.

All these demands show that creating maps is not only a matter of cartography any more and some technological shift is required. In last years, one can recognized the shift from the field of cartography to computer science with need for automated map generalization methods, where cartographic quality is one of many demands. Inspired by Mackaness et al. (2014), we can observe two following strategies to provide maps created by automated generalization process:

- The first approach is based on a complex automated generalization model in order to achieve high levels of automatization in the generalization process, see Figure 2.1. This approach is often applied by National Mapping Agencies (NMAs). They provide various topographic and thematic solutions from a single, highly detailed database. Most of the time the process contains tailor-made solutions to acquire ideal result for their production line. Usually, the result is a map of fixed scale with all cartographic aspects produced by the same well-know process as in the past. The only difference is the medium in which this is happening.
- The second uses more rigorous generalization approach. It is often used by web based mapping services, for example, Google maps, Bing maps and OpenStreetMap. They are based on scale dependent rendering where a simple filtering mechanism based on the level of detail and the entities' attributes take place. The large amount of zoom levels avoids the need for perfect legibility, precision, completeness and accuracy at each level compare to NMAs approach. The fact that user can zoom in and out so easily enables them to resolve any ambiguity at the smaller scale (Mackaness et al., 2014, p. 8), *e. g.* when the text is illegible at one level user simply zoom in, in order to read what is written. The map generalization for this approach is driven more by technical solutions from the field of computer science, computational geometry and others rather than cartography. The cartographic expertise is complemented by user testing. *i. e.* if the statistically significant group of people likes the resulting map then it is published, even if the result is not correct based on cartographic rules.

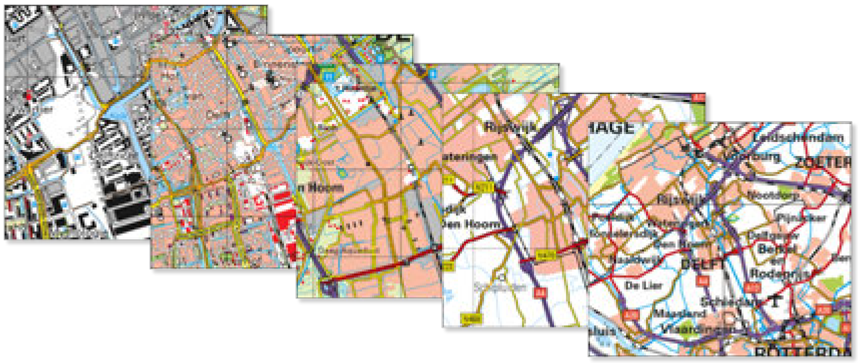


FIGURE 2.1 The Netherlands' Kadaster topographic map series.

Both of these approaches in the generalization community have strong arguments pro and con. Both also reflect two main approaches as to how the generalization is carried out. In addition to that, it brings interesting dilemma for map development, which can be phrased as the following question:
Should the result of map generalization be published only when the (high/traditional) cartographic quality is met?
The answer is not straightforward and it differs based on the approach. The first, complex solutions, mostly applied by cartographic experts by education, are commonly the employees of NMAs. They could argue that new development should be released if and only if the results are cartographic perfect in all aspects. This is valid argument but it

implies that any technological shift costs more, it is more complex to implement and maintain (Mackaness et al., 2014).

On the other hand, the second approach prefers the technological shift even with the drawback in form of lower cartographic quality. It is mostly supported by computer scientists, programmers or vendors. Therefore, it shares the same way of thinking (similar business model) as any software developing project; new features are integrated in smaller updates for direct application in practise. The cartographic quality is often derived from usability studies. In such a way, it easier to provide more up-to-date data and development is faster, because of the short developing cycle.

From the current state-of-the-art indicated above it is not clear which approach can best provide an ultimate generic solution for automated generalization in digital environment. Therefore, it is important to study, explore, compare and develop automated generalization solution to realize paradigma shift. Our vario-scale project explores new possibilities and realization for technological shift in map generalization. We do so by developing of specific data structure in computer world. Since we have limited resources the development is carried out in small steps, build on top of the current knowledge, consequently we share the view of the second approach. We use short development cycles to develop and to extend our knowledge. In more detail (see Section 1.5), we use simple development methodology; we develop theory, make a solution, test the solution against real world data, validate the results, adjust the theory and draw conclusions. This is a continuous iterative process until the result of development is sufficient. The same way of thinking is also reflected in the text of this thesis.

However, it would be mistake completely forget about the key players in the field of map generalization; the NMAs. The generalization was for long time their domain and they have drastically developed in last years and a lot of interesting work has been done there recently. Which of many of the proposed generalization operations and algorithms may be used in setting of continuous generalization, hence recent development in NMAs, will be the focus in Section 2.2. On the other hand, some continuous generalization functionalities are being made available to the user of GIS software, mobile application or web; and are extensively researched by other researchers. Therefore, we will explore those in Section 2.3 The key challenge in such a environment is zoomability. Section 2.4 explores the problem of integration map layers in zoomable user interfaces.

§ 2.2 Current development in National Mapping Agencies

Map generalization was driven by the needs of National Mapping Agencies users and it is still an important aspect of their work. NMAs face challenges because quite often their resources are decreasing and more frequent map updates at the various scales are expected. They are forced to use fewer personnel and shorter budget, but derive same results with less resources (Mackaness et al., 2014). This condition makes an ideal environment for automation, see an example in Figure 2.2. Therefore, we researched the current state-of-the-art of representative NMAs and their production with focus on automated generalization.

FIGURE 2.2 Workflow model for automated generalization of 1:50k map, Swisstopo (taken from (Käuferle, 2015)).

There are events such as The International Cartographic Conference, every two years and the annual ICA Commission of Generalisation and Multiple Representation workshop. Those events are places where practitioners from NMAs, researchers, developers and vendors meet and exchange their experiences and informations. There were also two other important events, two years apart, organized by the ICA Commission on Generalisation and Multiple representation and the EuroSDR Commission on Data Specification, under the theme 'Designing MRDB and multi-scale DCMs: sharing experience between government mapping agencies' (ICA and EuroSDR, 2013, 2015). The first symposium took place in Barcelona in March 2013 attended by 12 MNAs in total. The second in December 2015 in Amsterdam where 18 were represented.

The purpose of the symposium for NMAs was to learn from each other's experiences and to identify common needs and challenges that could be passed to industrials on the one hand and researchers on the other hand. The output of these events workshop report – Stoter et al. (2016), together with book chapter by Duchêne et al. (2014) and extensive analysis by Foerster et al. (2010) were used as input for this section.

Duchêne et al. (2014), Stoter (2005) and Foerster et al. (2010) identified the following steps in the introduction of automated generalisation in NMAs:

I. renewing data models (from CAD-like 'Map databases' to structured geographic databases, with a consistency between different levels of details),
II. designing the conceptual architecture (deciding what databases are derived from what data sources),
III. implementing generalisation processes (that actually perform automated derivation between data sources), and
IV. managing relationships between scales.
V. assessing the quality of the results.

For this section the most important is the third step; implementing generalisation processes (that actually perform automated derivation between data sets). It is most complex and technologically the most difficult one. While others have been already finished

TABLE 2.1 Progress of full automated generalization in NMAs' production for selected years. Year 2010 was surveyed by Foerster et al. (2010). Duchêne et al. (2014); ICA and EuroSDR (2013) covered results of 2013. 2015 is based on ICA and EuroSDR (2015); Stoter et al. (2016). Symbol × means that NMA was not surveyed in (Foerster et al., 2010) nor presented at a symposium.

NMA	2010	2013	2015
Belgium - IGN	no	no	no
Catalonia - ICC	no	no	no
Denmark - GST	no	no	no
Finland - NLS	×	1:100k, 1:250k	1:100k, 1:250k
France - IGN	no	'light' 1:25k	1:25k
Germany - AdV	no	no	no
Great Britain - OSGB	no	'light' 1:25k	1:25k
Ireland - OSI	no	no	no
Israel - SOI	×	×	no
Netherlands - Kadaster	no	1:50k	1:50k
Norway - Kartverket	×	×	no
Poland - GUGiK	×	×	1:250k
Spain - IGN	×	no	no
Sweden - LM	×	×	no
Switzerland - Swisstopo	no	no	1:10k, 1:25k, 1:50k
The Czech Rep. - ČUZK	×	×	no
Turkey - HGK	×	×	no
USA - USGS	×	no	no

some time ago such as the first step done for all NMAs in 2010 (Foerster et al., 2010), the third step is still in progress for most of the NMAs.

Table 2.1 gives an overview of automated generalization for individual NMAs in recent years. Even though the majority of NMAs apply some degree of automation already we consider only the production lines which are done in an automated way, *i. e.* only fully automated end-to-end solution are considered. There is a visible and significant shift from "full automated generalization process do not exist" in (Foerster et al., 2010) to semi-automated or fully automated process up to now. In 2010, the NMAs of Catalonia, Denmark, Germany, France and Great Britain have made major steps towards automated generalisation by adjusting available software or developing their own algorithms. Foerster et al. (2010) concluded that "Human interaction will always be required to improve the automated results and on-the-fly generalized datasets are not considered to be realistic."

However, only few years later, major steps in automated generalization process have been achieved. The first symposium (ICA and EuroSDR, 2013) revealed that eleven out of twelve NMAs present at the symposium have implemented automated or semi-automated solutions. This was first time that a full automated generalization process were applied, namely based on (Duchêne et al., 2014):

- OSGB Great Britain - 'light' 1:25k map derived from a mixed 1:25k-1:10k Digital Landscape Model (DLM)
- Kadaster Netherlands - 1:50k derived from 1:10k DLM

- IGN France - ‘light’ 1:25k derived from 1:10k DLM

The term ‘light’ means that the resulting map is not the usual high quality topographic map, but a lighter backdrop map, designed to be used at scales around 1:25k for overlaying other data onto it (Duchêne et al., 2014, p. 199, 385).

Note that these fully automated products were achieved while accepting compromises in terms of cartographic quality and differences compared to existing manually derived products (Duchêne et al., 2014, p. 382).

The trend of automation continued in following years. In 2015, there were six of eighteen NMAs carrying out fully automated workflow in their productions. The form of automation varies from NMA to NMA. Some of them have small-scale-generalisation processes implemented for years (Poland, Finland) and they are now developing towards the design of a large scale automated generalisation (Stoter et al., 2016). Others NMAs starting with automatization in their production lines tend to start with automation of large scale databases (10k) to medium-scale databases (50k). Stoter et al. (2016) say that some of the NMAs have implemented fully automatic procedures while others aim to follow within the near future, such as Ireland or Sweden. Many others have automated parts of the generalisation workflow. Based on (Stoter et al., 2016) Most of the NMAs that have implemented semi-automated workflows planned to replace these by fully automated workflows within the next two years (2016-2018).

Some of them are extending their production to more map scales, *e. g.* Netherland’s Kadaster the map of 1:100k derived from 1:10k. Note that such an extension is less complicated than full development because most of the tools can be reused or adjusted, *e. g.* a similar data model can by used, the same domain partition can be applied or identical cartographic operators can be performed.

§ 2.3 Continuous experience

Automated map generalization has been an important research area for years, descriptions can be found in multiple textbooks such as (Buttenfield and McMaster, 1991; Lagrange et al., 1995; Weibel, 1997; Mackaness et al., 2007). They provide comprehensive overview of many important aspects of map generalization. In parallel to this, there were always desires to use the new potential provided by digital environment. First, only the concepts and ideas were present in theory. However, technological advancements have led to maps being used virtually everywhere such as on tablets and smart phones, which leads to some implementations and real applications. Map use is more interactive than ever before: users can zoom in, out and navigate on the (interactive) maps. Recent map generalisation research shows a move towards continuous generalisation, without fixed target scales, and where smooth transition is applied. This is in contrast to ‘traditional’ generalisation of predefined scale-steps.

The issue of the continuous scale change has been quite extensively investigated; van Kreveld (2001) focuses on analysing the different ways of visually continuously changing a map, defining a number of operators that can be used. His work is based on transitional maps (maps that connect different predefined scales) and techniques of cartographic animation (Robinson et al., 1995; MacEachren and Kraak, 1997), such as morphing and fading.

Cecconi et al. (2002); Cecconi (2003) assume that corresponding objects of the different LODs are linked together, and the user with the help of on-the-fly generalization could select an intermediate scale between them. In this way, the perceived generalization can be understood as an ‘interpolation’ (or morphing) process between two different geometries. They also analyze the scope of the applicability of the generalization operators over the desired range of scales. They investigated the limits of applicability, where the ‘regime’ of an operator changes. Even though they study multi-scale solutions only, some concepts are valid for our solution as well, because they consider representation as a continuous function of scale, which has parallels with our approach.

Sester and Brenner (2005) demonstrate the gradual change of objects as a decomposition into a sequence of elementary steps. Later, any desired generalization level can be easily obtained by applying the appropriate sub-part of the sequence, see Figure 2.3. The steps are generated by unified operations similar to Euler’s operators (Eastman and Weiler, 1979) in such a way that they can be easily applied in the reversed order as well. They call this continuous generalization and so far it has been applied only for buildings. The method can be also used for incremental transmission of maps through limited bandwidth channels.

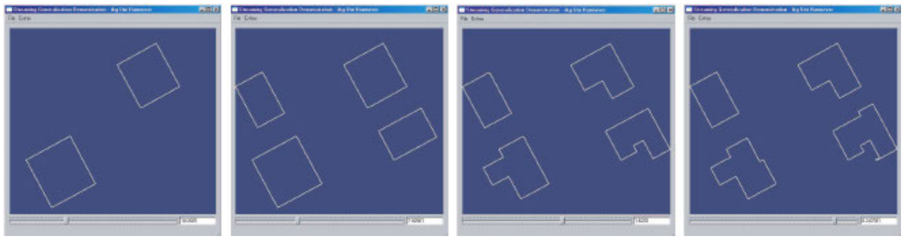


FIGURE 2.3 A sequence of operations in the inverse generalization process. More buildings details appear throughout progressive visualization of four levels of detail (taken from (Sester and Brenner, 2005, p. 7)).

Danciger et al. (2009) introduce deformation of the shapes of regions in a map during a continuous scale change, see Figure 2.4. They define mathematical functions for area/polygonal objects. However, the geometry forming a complete subdivision of space (a planar partition), which is important for vector map data, is not considered in this work.

Nöllenburg et al. (2008) give interesting examples of smooth transition for linear features between their representations at two scales. They focus on situations in which generalization operators like typification and simplification are not handled well. One such a example is replacing a series of consecutive bends by fewer bends. They attempted to cope with such cases by modelling the problem as an optimal correspondence problem between characteristic parts of each polyline. This presents an characteristic example of research in map morphing.

Brewer and Buttenfield (2007); Touya and Girres (2013) describe an interesting tool called ScaleMaster, which supports automatic multi-scale generalization. It is based on the model that formalizes how to generalize map features from different datasets through the whole range of targeted scales. Despite the fact that the tool focuses on generating a multi-scale/multi-representation solution, the idea of defining generalization actions for a range of map scales is an important concept.

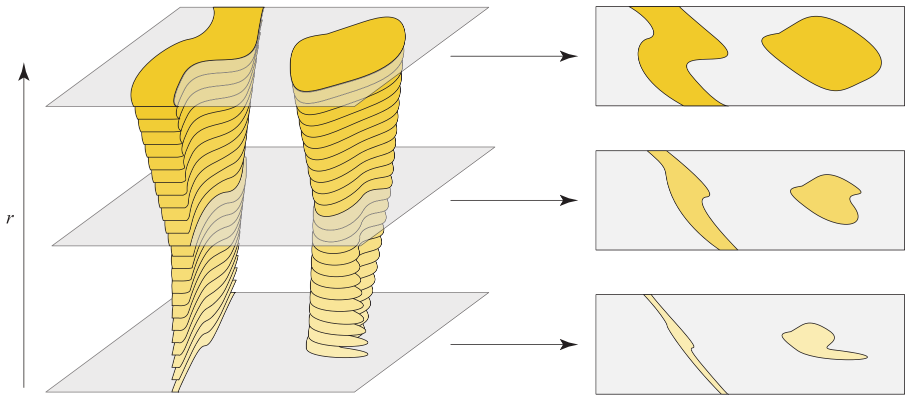


FIGURE 2.4 Deformation of two regions over time / scale (taken from (Danciger et al., 2009, p. 5)).

Chimani et al. (2014) apply a method where they remove edges of the road network map one by one. Therefore, the map is getting gradually simpler and simpler. There are almost unlimited numbers of possible orders in which edges can be removed. Therefore, they try to define the sequence of removing edges that gives the best result, while preserving graph connectivity. The best simplified map is the map with minimal change in connectivity. The method tries to minimize the sum of all differences for individual simplified maps, similar to the principle of the least square adjustment, where the minimized total of all changes is the optimal solution. To compute all possible permutations, they used linear programming. This is very expensive but gives an optimal solution which can be used as a benchmark. They then developed two novel heuristic optimization algorithms and compared them with the benchmark linear programming solution. They compared how well the two different algorithms approached the benchmark. It is one of the first papers to focus on global criteria during the continuous generalization; however, the quality of the generalization is still problematic. It shows that connectivity by itself is not a sufficient criterion for a good road network generalization result. There are aspects such as relative data density which should be considered. They faced a problem that a road segment can go missing or that a part of the network does not always nicely span the map extent, leaving large parts of the map empty. Furthermore, the overall impression of the map, where large rural and small urban areas should still be recognizable in a later stage of generalization, is still an issue.

§ 2.4 Generating intermediate scales

Interesting aspects have appeared due to wide spread digital map environment. The available cartographic data are usually based on a vectorial abstraction, which are well-suited for various representations. The fact that we can zoom freely into a such map leads to question as to which level of detail should be used in the depiction. Most systems rely on a limited set of maps, where those depicting a higher level of abstraction exhibit fewer details. This is especially true for the maps that were originally designed for paper medium. For instance, the Netherlands' Kadaster produce and provide on-line topographic vector data and raster maps at the scales of 1:10k, 1:25, 1:50, 1:100k, 1:250k, 1:500k and 1:1 000k (Stoter et al., 2014), see Figure 2.1.

The maps originally designed for an interactive virtual environment generate more abstraction levels, for instance, up to 19 zoom levels for Google Maps[1], Open Street Maps[2] and 22 levels for Bing Maps[3].

However, even with more LODs present it may lead to inappropriate data representation between two adjacent abstraction levels, see Figure 2.5

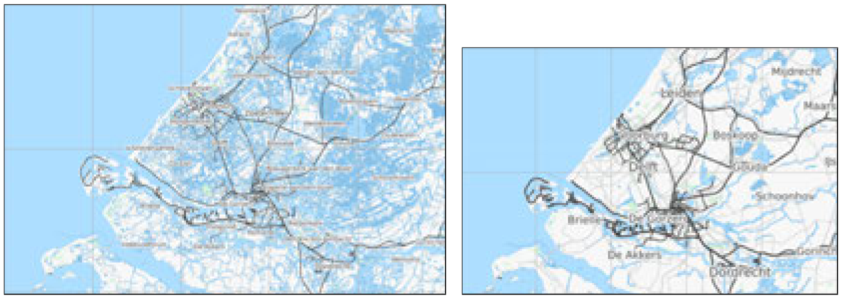


FIGURE 2.5 An example of map fragments at two consecutive scales (left at a larger scale, right at a smaller scale) with big differences in content (transport map from OpenStreetMap).

It is important to point it out that typically the work around is based on techniques – using visual effects such as blending between levels, or nearest neighbour scale selection. Blending can lead to ghosting artefacts or, for some applications such as surveying and topographical base map, to badly defined interpolations. Nearest-neighbour switches can be distracting and can be disturbing as associations between different levels might be unclear. These techniques are extension build on top of the solution and they only provide 'illusion' of gradually changing map content. Nevertheless, they are also part of the solution, because they have significant effect on users' impression and will be covered more in detail in Chapter 6. Altogether, if we want to provide ac-

1 https://developers.google.com/maps/documentation/static-maps/intro
2 http://wiki.openstreetmap.org/wiki/Zoom_levels
3 https://msdn.microsoft.com/en-us/library/bb259689.aspx

curate and clear cartographic data throughout the scales more explicit intermediate data/layers are desired.

Exploration work at IGN France (Dumont et al., 2015, 2016) has started to formulate an automated generalisation workflow for producing intermediate scale maps in a multi-scale pyramid to overcome gaps between different LODs. The additional scales are intended to eliminate or reduce user confusion caused by large scale jumps between maps and shock when zooming. They use already known matrices such as analytical measurement of the readability proposed by (Harrie et al., 2015) to identify if and where should be intermediate scales generated by automated generalization.

Bereuter et al. (2012) has focused on development of mobile map applications which suffer the same problems (number of map layers available). In addition to this, mobile maps suffer from the limitation of the screen size, especially for the display of overview information. They present a solution where the base map (or background map) is not strictly tied to the foreground data (*e. g.* POIs) as is usually the case. As a consequence, they change the assumption in map generalisation that the level of detail of the map background and foreground should always correspond, and thus change is in synchronicity across scales.

In their solution, users may adapt the degree of abstraction on a map of a specific scale depending on the usage scenario without changing the map extent. It is based on the change of the number of foreground objects displayed for a given LOD; and the change of the objects details, *e. g.* how dense the represented information is, in spatial and thematic terms.

§ 2.5 Future challenges

Section 2.2 has demonstrated that there are full automated or semi-automated map generalization solutions for the majority of NMAs nowadays. Table 2.1 has shown significant technological shift from 2010 to 2015 where more and more NMAs applied automated generalization. We can only assume that the trend; "to produce the map faster with fewer person and less financial resources", will continue.

While most NMAs have implemented a certain form of automation in their workflows, the development still focuses on producing maps at fixed LODs. Section 2.4 pointed out that generating more content or more intermediate LODs is needed for zoomable maps in web environment. It also presented researches where generating intermediate scales was addressed. Section 2.3 showed that automated generalization were intensively investigated by many researchers in order to find a good generalization solution in a digital environment. It indicates development shifting towards more smooth, continuous solutions in generalization, which could provide desired content for web maps.

This chapter introduced two things; First: how drastically and rapidly developing the field of map generalization is. Second: even though the automated generalization has been investigated extensively no total solution for web maps is known and the research and development is ongoing, especially towards continuous generalization. Therefore, different solutions such as vario-scale approaches are important topics for future.

3 Vario-scale data structures

The previous chapter presents state-of-the-art in map generalization at NMAs' and continuous generalization. There is a noticeable technological shift towards continuous generalisation which supports interactive map use where users can zoom in, out and navigate more gradual way. Despite some research efforts there is no satisfactory solution yet. Therefore, this chapter introduces the truly smooth vario-scale structure for geographic information where a small step in the scale dimension leads to a small change in representation of geographic features that are represented on the map. With this approach there is no (or minimal) geometric data redundancy and there is no (temporal) delay any more between the availability of data sets at different map scales (as was and is the case with more traditional approaches of multi-scale representations). Moreover, continuous generalisation of real world features is based on the structure that can be used for presenting a smooth zoom action to the user.

More specific, Section 3.1 and 3.2 provide historical overview of the development and the theoretical framework for vario-scale representations: the tGAP-structure (topological Generalized Area Partitioning). Section 3.3 describes the initial effort to generate the better cartographic content; the concept of constraint tGAP. Section 3.4 explains the 3D SSC (Space-Scale Cube) encoding of 2D truly vario-scale data. Section 3.5 shows idea how to combine more level of details in one map. Section 3.6 summarizes the open questions of the vario-scale concept and it indicates research covered in following chapters. Finally, Section 3.7 presents vario-scale data research in parallel to this PhD for progressive data transfer. Then, Section 3.8 summarises the chapter.

Own publications

This chapter is based on the following own publications:

- van Oosterom, P., Meijers, M., Stoter, J., and Šuba, R. (2014). *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, volume 2014 of *Lecture Notes in Geoinformation and Cartography*, chapter Data Structures for Continuous Generalisation: tGAP and SSC, pages 83–118. Springer International Publishing.
- Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014a). An area merge operation for smooth zooming. In Huerta, J., Schade, S., and Granell, C., editors, *Connecting a Digital Europe Through Location and Place*, Springer Lecture Notes in Geoinformation and Cartography, pages 275–293. Springer International Publishing. ISBN: 978-3-319-03611-3.
- Huang, L., Meijers, M., Šuba, R., and van Oosterom, P. (2016). Engineering web maps with gradual content zoom based on streaming vector data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:274 – 293.

§ 3.1 History of development

Data structures for multi-scale databases based on multiple representations attempt to explicitly relate objects at different scale (or resolution) levels, in order to offer consistency during the use of the data. The drawbacks of the multiple representations data structures are that they do store redundant data (same coordinates, originating from the same source) and that 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 (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 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., 2010; 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 and 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 LINE FEATURES: 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 (Günther, 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 but can be combined with several other line generalization algorithms. Note that these line data structures can only be used for spatial organization of single objects and for (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 organization (indexing) and multiple level of details. Its main disadvantage, however, is that it is a static structure. The first dynamic vector data structure supporting spatial organization 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 object 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.

GENERALIZED AREA PARTITIONING: The BLG-tree and Reactive-tree structures are not well suited for an 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, 1995). The BLG-tree, Reactive-tree, and GAP-tree data structures can be used together, while each supports different aspects of the related generalisation process, such as selection and simplification, for an area partitioning (van Oosterom and Schenkelaars, 1995).

Following the conceptualization 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 – redundant storage and slivers near the boundary of two neighbour areas.

§ 3.2 GAP tree background

The first tree data structure for generalised area partitioning (GAP tree) was proposed by van Oosterom (1995). 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; see Figure 3.1. This principle has been applied to the Digital Land Mass System-Digital Feature Analysis Data (DLMS DFAD) data structure (DMA USDMA, 1986), because it already had this type of polygons organization. When tested with the Reactive tree and the BLG-tree, it was possible to zoom in (zoom out) and obtain map representations with more (less) detail of a smaller (larger) region in constant time (see Figure 3.3, 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 3.1 gives a schematic representation of such a GAP tree.

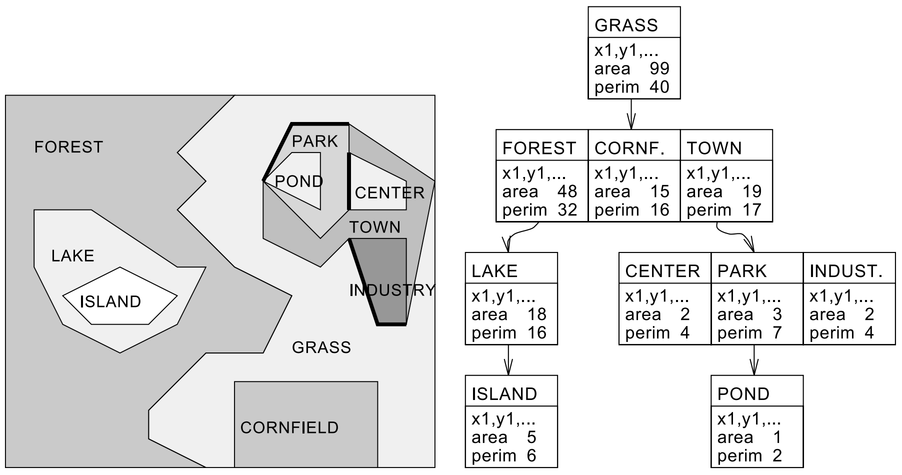

FOREST
PARK
POND
CENTER
TOWN
LAKE
ISLAND
INDUSTRY
GRASS
CORNFIELD
GRASS x1,y1,... area 99 perim 40
FOREST x1,y1,... area 48 perim 32
CORNF. x1,y1,... area 15 perim 16
TOWN x1,y1,... area 19 perim 17
LAKE x1,y1,... area 18 perim 16
CENTER x1,y1,... area 2 perim 4
PARK x1,y1,... area 3 perim 7
INDUST. x1,y1,... area 2 perim 4
ISLAND x1,y1,... area 5 perim 6
POND x1,y1,... area 1 perim 2
(a)The scene (b)The GAP tree

FIGURE 3.1 The original GAP tree (van Oosterom, 1995).

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; e.g., 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; Figure 3.3, 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) neighbour areas as well. Both suggestions are based on an analysis using a Triangular Irregular Network (TIN) structure.

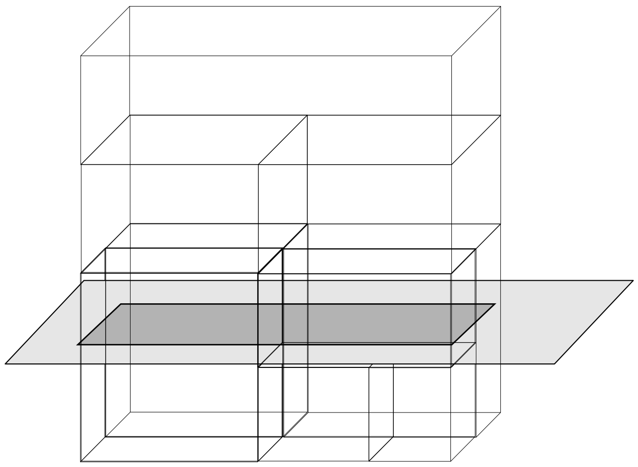


FIGURE 3.2 Importance levels represented by the third dimension (Figure 3 of (Vermeij et al., 2003)). At the most detailed level (bottom) there are several objects, while at the most coarse level (top) there is only one object. The hatched plane represents a requested level of detail, and the intersection with the symbolic 3D volumes then gives the faces.

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 extended by the importance value range (on the z-axis) for which it is valid (see Figure 3.2). 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 data structure which also avoids the redundant storage of geometry at different levels of detail (in the edges). Figure 3.4 shows the generalisation process for the tGAP structure in which also a simplified version of the edges is available, without explicitly storing them.

MORE COMPACT TGAP STORAGE: The structure stored the node and face data very efficiently. However, there was a lot of redundancy in the way the edges were stored. The edge table for a realistic data set did have up to 15 times more rows than the original edge table (and the theoretic worst case is even $O(n^2)$ with n the number of edges at the largest scale). This was mainly due to the changing references to the left and right faces after merging two neighbour faces. Therefore, Meijers et al. (2009) have proposed the modification of structure which removes this redundancy without loss of functionality.

It was based on two following aspects; First, multiple edge rows in the edge table may merge in one row if 1) they relate to the same edge and 2) only the left/right references were changed (not the edge geometry). This results in no change for the geometry, start and end nodes, and id attributes. The imp_low and imp_high attributes con-

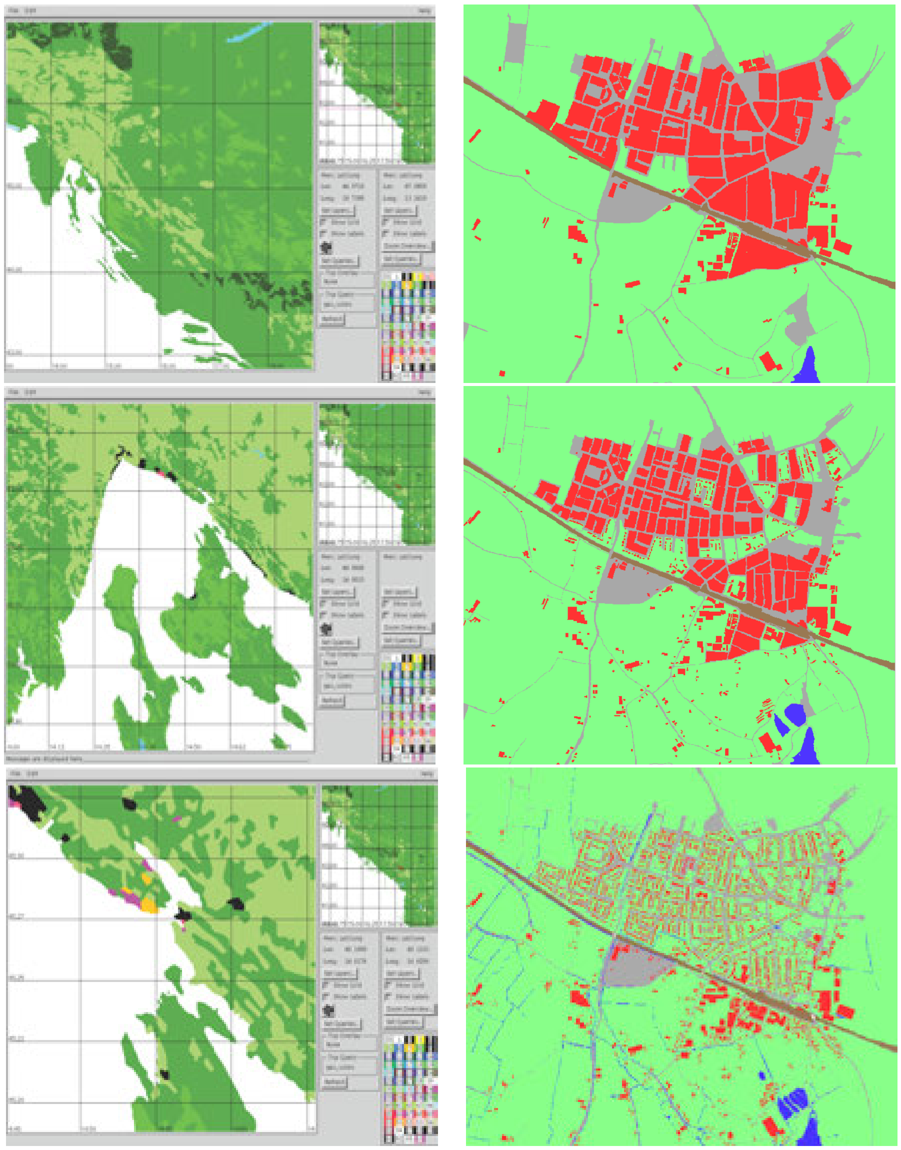


FIGURE 3.3 Left: GAP-tree principle applied to DLMS DFAD (add detail when zooming in). Right: GAP tree applied to large scale topographic data set (shown at same scale). Taken from (van Oosterom, 2005).

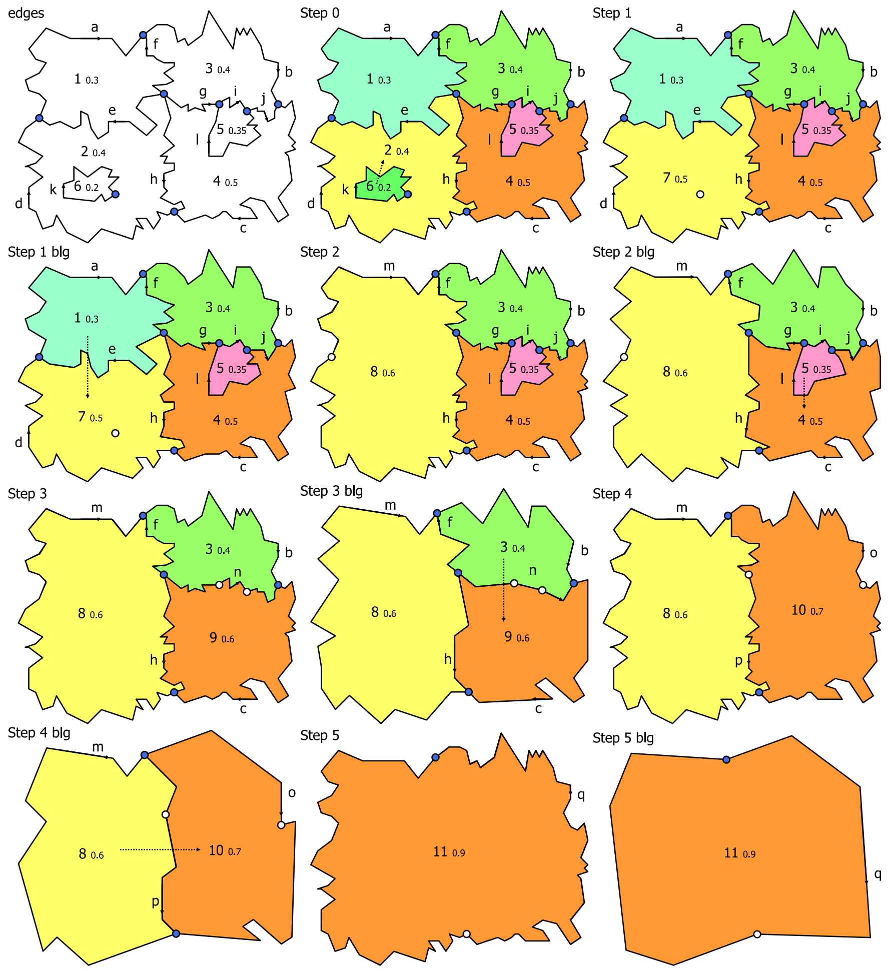


FIGURE 3.4 Generalisation example in five steps, from detailed to course, where merge and simplification operations used. Steps with *blg* in the title shows the effect of simplifying the boundaries via the BLG tree. Note that nodes are depicted in blue and removed nodes are shown for one next step only in white. Taken from (van Oosterom, 2005).

tain the union of all importance ranges of the edge (which are per definition adjacent ranges). Second, the left and right face references should be stored related to the lowest importance range. In these cases the referred face (and the corresponding edge) with the highest imp_low level is used as start in the tGAP face-tree and the tree is traversed upwards until the face identifier at the right imp level is found. The left/right information and the tGAP face-tree can then be exploited to properly identify the areas at a certain importance level (scale).

Then, just storing only rows for edges that are really new (because these edges are merged) saves a lot of storage (rows) (the number of rows reduced by a factor of at least 2 as edges are merged pairwise) (Meijers et al., 2009, p. 14). Saving storage space also implies saving data transfer times in a server-client architecture.

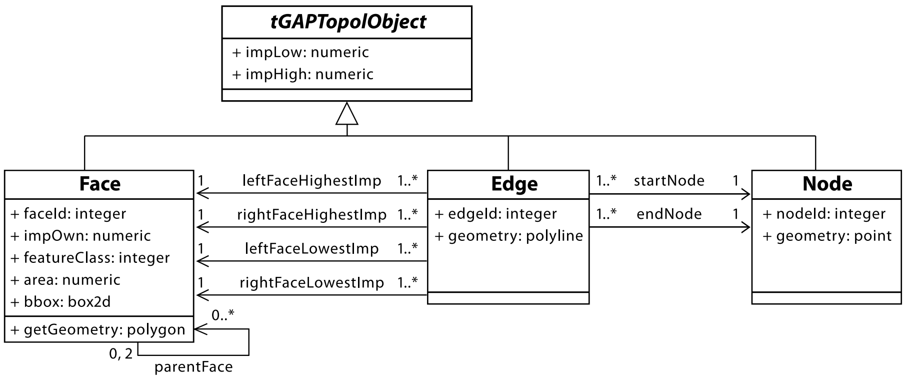
DESCRIPTION OF THE STRUCTURE: All modifications resulted in the tGAP structure captured in Figure 3.5. The figure summarizes the structure in current implementation, its database tables and UML diagram.

The description originates from (Meijers, 2011, p. 157); "Edges table (polylines) is the most important in the database. It contains edges, The imp_low and imp_high attributes define the range of map scales for which a topological entity is valid and has to be shown. Face references that are stored are limited to the neighbours that are adjacent at the start of such a scale range ($left\_face\_lowest\_imp$ and $right\_face\_lowest\_imp$) and which faces are neighbouring at the end of this range ($left\_face\_highest\_imp$ and $right\_face\_highest\_imp$). This way it is prevented that a lot of duplicate edge records have to be stored, while the only thing that changes (due to a generalisation operation) is a neighbouring face. Note that the two extra face pointers (at the high end of the scale range) are added, as this is useful for replaying generalisation operations progressively, while traversing the tGAP structure from top to bottom (in a strict sense, these extra pointers are not necessary, but this takes away the necessity to perform a translation of the face pointers using the tGAP face tree hierarchy to the high end of the scale range, before sending the edge)."

§ 3.3 Improved cartographic quality in constraint tGAP

The tGAP (topological Generalised Area Partitioning) structure was designed to store the results of the gradual generalisation process as described in previous sections. Earlier research had been focused more on proofing the concept, designing the solution, and shifting the technical limitation than improving cartographic quality of the tGAP. The generated content had been steered by an *importance* function, for selecting the least important object and then merge with the most compatible neighbour based on a *compatibility matrix*. From the cartographic point of view this solution was not really optimal.

The initial attempts for improving cartographic quality were carried out by Haunert et al. (2009); Dilo et al. (2009). Figure 3.6 presents an approach, named *constrained tGAP*. Instead of generating new maps from one source (Figure 3.6a) or a sequence of simpler and simpler map as in classical tGAP (3.6b), the process is based on a reference, smaller scale, well generalized map, see 3.6c. With the additional smaller scale input data (the constraints), the iterative algorithm can be controlled to obtain higher quality (intermediate) maps.



(a)UML diagram

```
CREATE TABLE tgap_faces (
face_id integer,
imp_low numeric,
imp_high numeric,
imp_own numeric,
feature_class_id integer,
area numeric,
bbox box2d);
```

(b)Face table

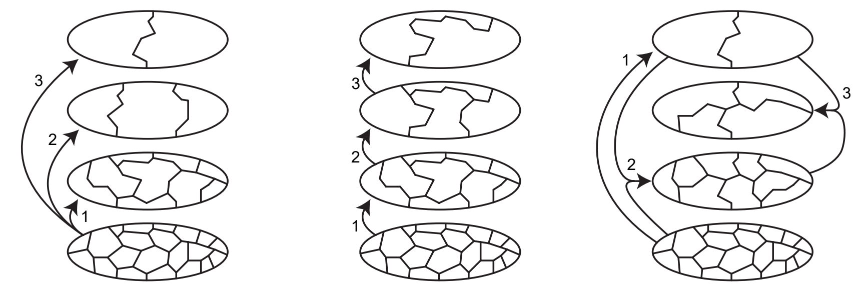
```
CREATE TABLE tgap_face_hierarchy (
face_id integer,
parent_face_id integer,
imp_low numeric,
imp_high numeric);
```

(c)Face hierarchy table

```
CREATE TABLE tgap_edges (
edge_id integer,
imp_low numeric,
imp_high numeric,
start_node_id integer,
end_node_id integer,
left_face_lowest_imp integer,
right_face_lowest_imp integer,
left_face_highest_imp integer,
right_face_highest_imp integer,
geometry geometry);
```

(d)Edge table

FIGURE 3.5 UML diagram and the database tables for tGAP, source: (Meijers, 2011, p. 159).



(a)Generalisation from a single source dataset (repeated optimization). (b)Successive generalisation (classic tGAP approach). (c)Intermediate representations fitting in the target representation (constraint tGAP approach)

FIGURE 3.6 Approaches to create a sequence of LoDs from (Haunert et al., 2009).

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.

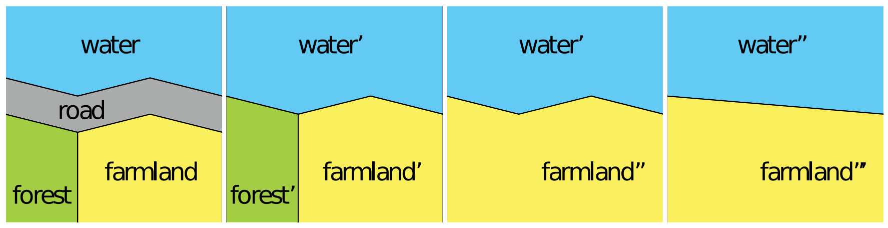
The result of generalization is highly dependent of the 'target' generalised dataset (smaller scale input) and it can be obtained via two routes: 1) from an external, independent source or 2) via a different generalisation algorithm (same source). The first option was explored by Dilo et al. (2009). The second option was investigated in (Haunert et al., 2009). Dilo et al. (2009) implemented constrained tGAP with real topographic datasets from the Netherlands for the large-scale (1:1,000) and independent medium-scale (1:10,000) data. This solution may suffer from geometrical and time inconsistencies, *e. g.* the geometry of reference scale is different or not up-to-date. This could make the preprocessing step even more complex.

On the other hand, Haunert et al. (2009) used German Land Cover data in 1:50,000 scale (ATKIS DLM50) as input. They eliminated the problem of inconsistency by deriving small scale data by the optimized (and computationally expensive) algorithm from the same base data.

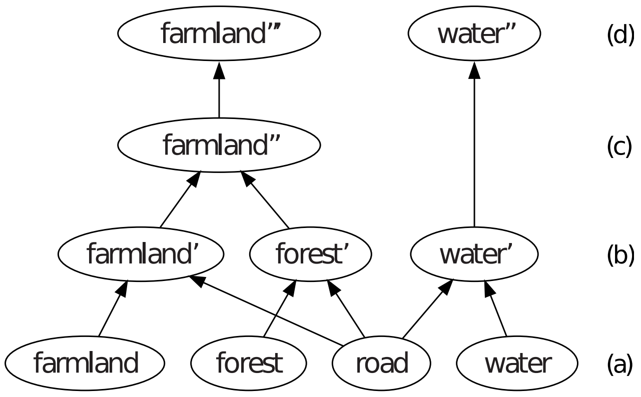
The most significant limitation of using constrained tGAP is in the fact of using additional knowledge derived from other solution. In theory, it means that the solution cannot run independently. Moreover, the results are very dependent on the 'target' small scale approach obtained from complex and expensive preprocessing step.

§ 3.4 The tGAP structure represented by the 3D Space-Scale Cube

The tGAP structure has been presented as a vario-scale structure (van Oosterom, 2005). So far, the historical overview of development and current implementation have been



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



(e)Corresponding tGAP structure

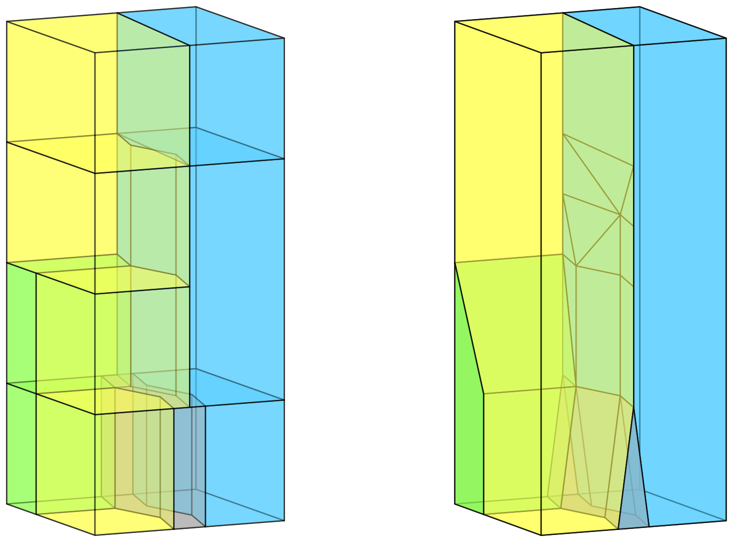
FIGURE 3.7 The 4 map fragments and corresponding tGAP structure (taken from (Meijers, 2011, p. 70)).

presented. Here, we will explain how more gradual (morphing-like transition) between the stages of the structure can be ensured.

The tGAP structure can be seen as result of the generalisation process and can be used efficiently to select a representation at any required level of detail (scale). Figure 3.7 shows four map fragments and the tGAP structure in which the following generalisation operations have been applied:

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

The tGAP structure is a Directed Acyclic Graph structure (DAG) and not a tree structure, as the split causes the road object to have several parents; see Figure 3.7e. In current tGAP 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 as a 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



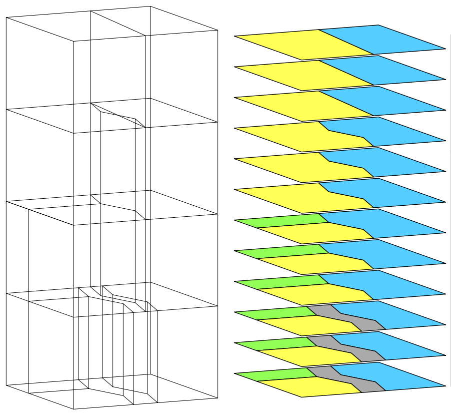
(a)SSC for the classic tGAP structure (b)SSC for the smooth tGAP structure

FIGURE 3.8 The space-scale cube (SSC) representation in 3D, source (Meijers, 2011).

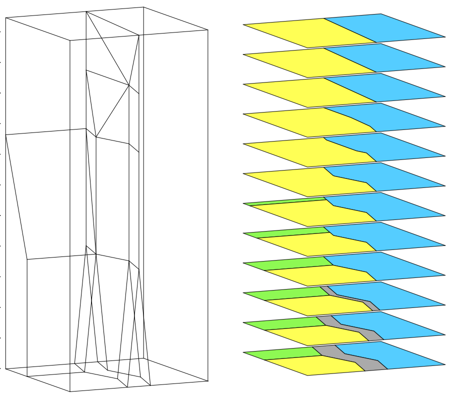
space-time cube as first introduced by Hägerstrand (1970). Figure 3.8a shows this 3D representation for the example scene of Figure 3.7. In the SSC the vario-scale 2D area objects are represented by 3D volumes (prisms), the vario-scale 1D line objects are represented by 2D vertical faces (for example 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.

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 generalisation actions approaching vario-scale, but not truly smooth vario-scale. Split and merge operations cause a sudden local 'shock': a small scale change results in a not so small geometry change where, for example, complete objects disappear; see Figure 3.9a. In the space-scale cube this is represented by a horizontal face; a sudden end or start of the corresponding object. Furthermore, polygon boundaries define faces that are all vertical in the cube, *i. e.* the geometry does not change at all within the corresponding scale range, 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 Figure 3.8b, results in *smooth tGAP* providing gradual transitions; see Figure 3.9b.

Then, a map can be obtained from this volumetric partition by horizontal slice. The continuously changing map one can imagine as gradually moving the slicing plane (and scaling) from the top of the cube downwards, there will be any object suddenly appearing or vanishing. All changes result in a smoothly changing 2D map: a small change in the map scale means a small change in the geometry of the resulting map.



(a)Wireframe and slices of classic SSC (with shocks).



(b)Wireframe and slices of smooth SSC.

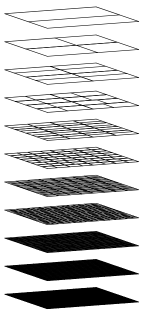
FIGURE 3.9 The map slices of the classic tGAP structure: (b) step 1 (collapse), (c) step 2 (merge) and (d) step 3 (simplify). Note that nothing changes until a tGAP event has happened. Figures originate from (Meijers, 2011).

§ 3.5 Mixed-scale representations

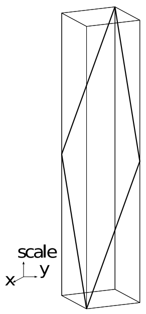
So far discussion has been restricted to the creation of 2D maps by slicing the SSC horizontally, however nothing prevents us from taking non-horizontal slices. Figure 3.10 illustrates a mixed-scale map derived as a non-horizontal slice from the SSC. What does such a non-horizontal slice mean? More detail 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 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 generalised (small scale) features far away and less generalised (large scale) features close to the observer.

The mixed-scale representation brings new aspects to consider:

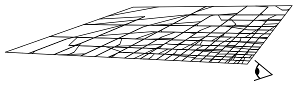
- *Non-planar slices* – 2D map can also be obtained by slicing surfaces that are non-planar; *e. g.* a bell-shaped surface that could be used to create a meaningful 'fish-eye' type of visualizations, see Figure 3.11.
- *Slices resulting in multiple parts* – It might be true that a single area object in the original data set might result in multiple parts in the slice (but no overlaps or gaps will occur in the slice). This increases the number of objects in the map and it results in a degradation of the quality of the map.
- *Limits of slicing surfaces* – What are other useful slicing surface shapes? Are there any limits? One can think of a bell-shaped, a sine curve surface or a 3D mesh object based surface. On the other hand, a folded surface seems to be nonsensical as it could lead to two representations of the same object in one map/visualization.



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

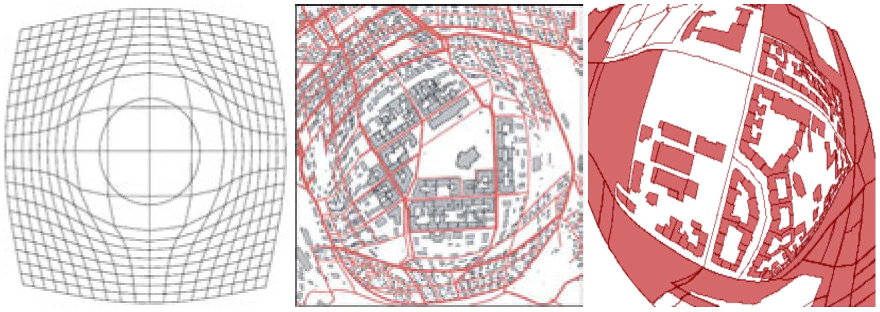


(b)How the non-horizontal slice of (c) is taken.



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

FIGURE 3.10 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). All figures are taken from (Meijers, 2011).



(a) (b) (c)

FIGURE 3.11 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).

§ 3.6 State of the art

The text above gave overview about past developments and the state-of-the-art of vario-scale concept when PhD project started. There were also some open questions which indicate what were the main limitation and desires at that time, from (van Oosterom, 2005; van Oosterom and Meijers, 2012; Šuba, 2012). Below a list of open issues prior to this research is included:

I. It will be necessary to verify theoretical concepts about SSC more practically. Further, to test whether it is necessary to create and store SSC as a 3D structure or is it possible to work with only the 2D representation? How to realize conversion from tGAP to smooth tGAP?

II. Explore new possibilities of creating maps by slicing. It should be possible to create a map with a slice which is not even horizontal. It would lead to a 'mixed-scale' map. How can this be done efficiently? How will users perceive this?

III. The current structure only explicitly supports area features. Line and point features are not yet included explicitly in the storage structure. However these type of objects are produced during the creation of the tGAP structure. The collapse process is a good example. When the long feature is collapsed to a skeleton it changes dimension, *e. g.* from dimension 2 to dimension 1, from an area feature to a line. It is convenient to store information about the collapsed feature.

IV. Labels are an important part of maps, but are not included in the structure. Objects in the map need a label and every label in the map takes space. It is a classical cartographic problem: each object that has to be labelled allows a number of positions where the corresponding label can be placed. However, each of these label candidates could intersect with other label candidates and other objects as well. It is possible to find the right solution for just one predefined selected scale, but how can we find the right solution for the variable-scale?

V. Larger real world data sets should be tested (to further assess the potential of vario-scale maps). How should we deal with millions of records not fitting into the memory? One solution could be split in smaller parts. After that every part has to be computed separately (and potentially processed in parallel) and in the end these have to be merged together. How can the whole process be made manageable?

VI. One merge operation at a time is supported in the structure. More generalization actions could run concurrently, *e. g.* a parallel merging process could be an improvement with objects involved at different locations.

VII. Make the structure more dynamic. The current tGAP structure is a static one. It cannot handle changes of source data over time. If a small change of data takes place in the structure, the whole structure has to be recomputed.

VIII. Better decisions, because the selection of an object is based on area and class of object only. Moreover the thematic semantic aspect should be considered because there is no approach that would take into account the feature type of the objects, *e. g.* linear or areal.

TABLE 3.1 The list of open questions from previous publications (August 2012)

number	Proposed concept	Implemented, tested during PhD
I	SSC - practical test, convertion	yes, Chapter 6
II	Slicing, 'mixed-scale' map	yes, Chapter 6
III	Other map features	yes, Chapter 4
IV	Labels	no
V	Bigger dataset	yes, Chapter 5
VI	More operations simultaneously	yes, Chapter 4
VII	Dynamic structure	no
VIII	Better decisions	yes, Chapter 4
IX	Better techniques for gen. map content	yes, Chapter 4
X	Progressive transmission	no, §3.7
XI	User testing	yes, Chapter 6

IX. Improve the current vario-scale techniques for better vario-scale map content. Can we come up with a better method to steer the generalization process that is used to obtain data for the tGAP data structures?

X. Explore progressive transmission (and implement a working prototype in a GIS viewer, such as QGIS). A streaming solution, in which previously sent ordered geometry is reused, will only be possible with an approach where redundancy is addressed from the start. Data streaming becomes more important with a very large dataset. Having a coarse overview first, adding gradually more detail, makes it possible to stop when sufficient detail has arrived.

XI. Measure the impact of vario-scale on users. Is the vario-scale approach perceived better as multi-scale solution or faster, more intuitive or more effective for user navigation. User testing is a tool for such a validation. However there is no prototype to realize such testing at this moment.

These open questions give us an idea what were the most urgent issues in the project. However, the number of problems faced was quite large and the research time was limited. Therefore we considered; 1) the original research proposal, 2) discussion with the STW Users Committee, 3) a logic order, and 4) urgency in the vario-scale research. Those aspects have been used for defining the main research parts in the next chapters. Table 3.1 summarize the list above. It also indicates what was selected to be researched in this thesis; If so, the related chapter is mentioned.

§ 3.7 Progressive data transfer

Table 3.1 indicates that topic *progressive data transfer* was investigated during the time of this PhD project. Even though the topic is not within main focus of this work, it is important to mention it because shows additional aspect of vario-scale approach and completes the picture of the project.

Huang et al. (2016) have shown a solution for the progressive transfer of geographic vector data using vario-scale data structure in web-services setting. Note, that no 3D SSC and slicing as described in Section 3.4 was implemented, it is based on 'normal tGAP'. It focuses on data streaming through possibly limited bandwidth. First, the

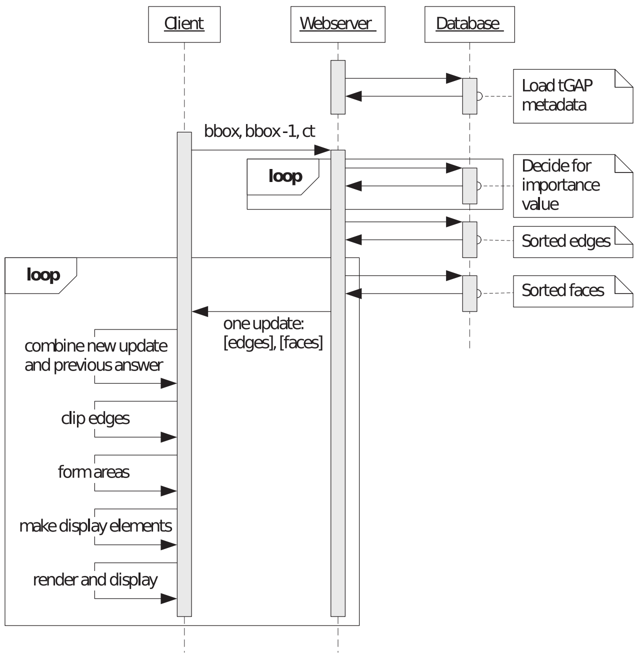


FIGURE 3.12 Client-server communication architecture for progressive data transfer (Figure 9 of (Huang et al., 2016)).

vario-scale data structure is prepared on the server side which encodes the results of the sequence of generalization steps; then, client-server communication architecture with client-side visualizations and smooth zoom interactions is efficiently realised using data from a vario-scale server.

In situation when the user performs an zooming operation, see Figure 3.12, the request parameters are; 1) a pair of bounding boxes (capturing the original and desired map state) and 2) optimal counts[1]. The response consists of a stream of chunks. Each chunk contains sufficient information (*i. e.* edges, faces and importance value) to bring the map at the client side to a new consistent state. Thus the map is gradually refined with content after an updated edges and face have been processed –- the user might still be zooming. Therefore, the response is dealt with chunk by chunk. For every incoming chunk of the response:

I. The client selects which edges and faces have to remain from the last response received.
II. The winged edge structure is updated accordingly (adding new edges/faces, removing unneeded faces/edges).

1 Optimal count defines the user preference of how much data is retrieved and visualised. By changing it, the user can modify the map appearance from a dense map to a map with few objects.

III. Ring and display objects are formed for updated topology primitives (faces) and the relevant styling is added.

IV. Display objects from the previous step are updated accordingly (those SVG polygons that are no longer needed are deleted and new ones that were formed are added).

This leads to faster visual feedback for a user. The user can see the map improve while downloading additional refinements (with a partial answer the map can already be constructed).

The idea of vario-scale as an optimal tool for progressive transfer of geographical data via the Internet was one of the original hypotheses, but it has been never proven until now. The paper by (Huang et al., 2016) demonstrated the potential of the vario-scale concept, because solutions for efficiently transferring vector data especially for limited bandwidth are urgently needed in desktop and mobile applications at high performance levels. To conclude, this makes the vario-scale approach relevant research in the field of automated map generalization because it combines 1) minimal duplication in data storage, 2) explicit links (because of the hierarchy structure) between the map objects, and 3) support for progressive data transfer.

§ 3.8 Final remarks

This chapter presented vario-scale development in the past and state of the art when PhD project started. It explained vario-scale approach in more detail using a specific data structure, which stores the results of map generalization actions; features are generalized in small steps, progressively leading to simpler and simpler maps. The main advantages of the concept compared to the multi-scale approaches can be summarized in the following aspects:

I. Redundant storage is eliminated as much as possible by avoiding duplication of features (as in multi-scale) and storing the shared boundaries between neighbouring areas instead of the explicit polygons themselves, *i. e.* using a topological data structure composed of nodes, edges and faces.

II. More intermediate representations can be constructed, because the automated generalization process records for every topological primitive (node, edge or face) the valid range of map scales for which this element should be shown.

III. The lineage of the generalization steps is stored explicitly and implies the links between generalized and original objects. Obtaining these links is regarded as a very difficult task for multi-scale databases. The main advantages of this are (1) making progressive updates/transmission possible and (2) future updates of the features are more easy to implement without re-generating the dataset.

IV. The Space Scale Cube (SSC) conceptual model makes possible really smooth transitions throughout the scales. It changes representation from 2D map features to 3D polyhedra. Then to derive a map from an SSC, the model is sliced by a plane at the desired level of detail. Hereby, exact LODs can be chosen and smooth transitions between abstraction levels are ensured.

Our approach can, in some aspects, already provide an advanced solution comparable to results of a well-known multi-scale solution. However, the vario-scale concept is still ongoing research which requires further development and testing. Therefore, Section 3.6 identified the most urgent needs of the approach and some of them will be addressed later.

For the remaining chapters it is good to keep in mind that vario-scale produce a sequence of successively more generalized maps, so that these maps go well together, similar to (Chimani et al., 2014); instead of considering each level of generalization independent. We aim at providing more degrees of freedom in data use. In our perspective, the sequence of generalization steps is more valuable than just the final map of specifically-identified scale. Our ambition is to produce data without explicit lower and/or upper bound. This makes the method very generic. On the contrary, it is difficult to compare with current maps because vario-scale is different by design.

4 Techniques for better vario-scale map content

The previous chapters covered the research where the vario-scale structure has been introduced. The main aim of the research was general functionality, performance and optimization. So far, the technical aspects had higher priority than the map content. Therefore, this chapter focuses on improving our development kit for generating vario-scale content. It presents a strategy to provide good cartographic results throughout all scales and properly stored in the structure. First, Section 4.1 specifies our target. Section 4.2 presents solutions of other researchers. Section 4.3 introduces concepts and tools which are used later in newly designed process. This is demonstrated on road features in Section 4.4. The section presents the generalization approach for the whole scale range from large scale, where roads are represented as area objects, to mid and small scales, where roads are represented as line objects. In our suggested gradual approach even for one road the representation can be mixed, both area and line, which may provide better transition phase and better impression for the user. This is shown in Section 4.5. Data density is still an issue, therefore, Section 4.6 suggests solution for preserving different feature density in the map, *e. g.* high feature density in the cities and sparsely occupied rural regions. Finally, Section 4.7 concludes the chapter and presents some open questions.

Own publications

This chapter is based on the following own publications:

- Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014b). Continuous Road Network Generalisation. In *Proceedings of the 17th ICA Workshop on Generalisation and Multiple Representation, Vienna, Austria, September 23, 2014*, pages 1–12.
- Šuba, R., Meijers, M., and van Oosterom, P. (2015). Large scale road network generalization for vario-scale map. In *Proceedings of the 18th ICA Workshop on Generalisation and Multiple Representation, Rio de Janeiro, Brazil, 21 August, 2015*, pages 1–10.
- Šuba, R., Meijers, M., and Oosterom, P. v. (2016b). Continuous road network generalization throughout all scales. *ISPRS International Journal of Geo-Information*, 5(8):145.

§ 4.1 Objectives

The vario-scale concept has a wide range of application from navigation software, through desktop GIS to mobile applications. Recent development was focusing on on-line applications with effective vector data transfer, see Section 3.7. It is based on the idea that if features are generalized in small steps, smooth transitions between subsequent object representations can be derived (Sester and Brenner, 2005; Midtbø and Nordvik, 2007; Chimani et al., 2014; Huang et al., 2016). This provides a different approach to conventional discrete scale maps on the Internet. Those discrete maps have abrupt changes between scales, which can lead to disorientation, result in confusion and eventually the frustration of map users. Figure 4.1 shows a detailed road network in one map scale in comparison to the sparse network at the next available scale.

The development of the vario-scale data structure reaches the state where we have sufficient tools to generate meaningful map content for tGAP structure. We have well-know remove/merge, collapse/split and line simplification operations. This gives us options to get reasonable map content. The main requirement of the process stays the same over the project research. We want to create meaningful, reasonable and 'cartographically correct' maps. In more detail all generalization actions during the tGAP creation process should lead to good quality results from the cartographic point of view, *i. e.* Every intermediate map gives a proper impression to the map reader. Nevertheless, the generation of meaningful output of continuous generalization is still a challenge.

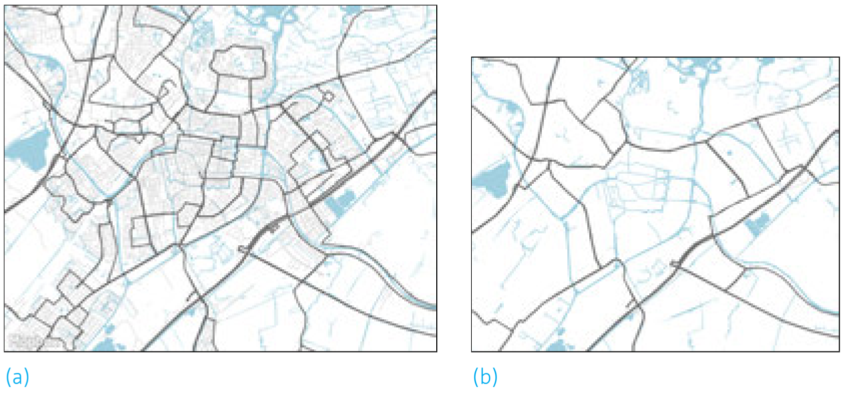


FIGURE 4.1 An example of map fragments at two slightly different scales; (a) at a larger scale, (b) at a smaller scale. Note that only the water and road network are displayed for these two scales, which were obtained directly from the original multi-scale database without any content modification (source: OpenStreetMap, styling: Mapbox Studio).

Our ambition is to develop a better strategy to create vario-scale representation. In this chapter, our working assumption is that by capturing the whole generalization process from large-scale input data, being progressively collapsed from areas (large scale) and/or merged into lines (small scale), it is possible to obtain a better representation with the following properties:

- Better preserved (road) network connectivity.
- Changes in geometry and/or classification are gradual.
- Small details are gradually eliminated (unimportant dead-ends (cul-de-sac) are removed and no new dead-ends are created).
- Road network semantics are taken into account, and the overall map impression is emphasized (remains during generalization).

This may give in theory a better impression to the user, *e. g.* while zooming in and out. Note that user testing should confirm that, and it will be investigated more in Section 6.5.

A property of the transition phase from large to small scale is that area and line representations are mixed together, *e. g.* second order roads at the same map scale may be represented partially by areas and partially by lines. Thus, by design our approach does not rely on homogeneous road segment representations. This is a non-trivial issue for conventional data structures and requires specific data modelling and modification of the data structure, both of which will be explained later in the text. In previous papers on vario-scale data using the tGAP structure (van Oosterom, 2005, p. 345) or (Meijers, 2011, p. 199), it has been mentioned several times (in the 'Future Work' section) that it should be possible and highly desirable to include features that have a line representation to improve the content of the map significantly. Now, for the first time, this is being realized and tested with real data.

To summarize briefly, our main goal is to generate good cartographic results representing all scales in the tGAP structure, to have better support and data content for smooth zooming.

To accomplish such a goal, we have the following requirements based on our vario-scale concept:

I. To introduce line feature representations.
II. To deal with mixed areas and line representations.
III. To generalize in small steps.
IV. To use an area partition as input.
V. To preserve the meaning of road network.

§ 4.2 Linear versus area representations / Related work, road network generalization

Our tGAP tools are in principle designed in a very generic way, however, in this chapter will be demonstrated mainly on road network data because the roads (similarly rivers and water channels) as linear/infrastructure objects are the 'backbone' of many map types. They improve the legibility of maps and help users to orient and recognize the depicted real-world situation more easily. On top of that, road network generalization forms a prerequisite for all other topographic generalization action (operators) and is thus a fundamental operation in the overall process of map and database production (Weiss and Weibel, 2014).

Various earlier attempts have explored ways of generalizing road networks as an important part of generalization process. The road processing has been extensively studied where at a given scale, two main representations of the roads may be found: linear and area representations. On large-scale maps, for Belgium, the Czech Republic, the Netherlands, the United Kingdom and other countries, a road segment is represented by area geometries. Together, the collection of road areas represents the road network. These areas form an implicit road network graph comprised of edges and nodes. At smaller scales, the road segments are represented by line geometries. These lines correspond more directly to the edges in the road network graph. At smaller scales (with road segments directly represented as edges), the road network generalization can have the emphasis on: (1) the linear road representation itself; or (2) the areas between the roads as regions bounded by minimal road loops or cycles, *e. g.* containing built-up area, forest or terrain. The linear emphasis approach considers the network as a set of linear connected elements, while the regions emphasis approach concentrates on the 'space between' the roads. Sometimes, the term 'area partition' (Edwardes and Mackaness, 2000) or 'mesh density' (Thomson and Richardson, 1999; Chen et al., 2009; Li and Zhou, 2012) is used. For smaller scale road network generalization, where road sections are represented as lines, both views (either linear or regions emphasis) have their own advantages and disadvantages (Edwardes and Mackaness, 2000).

Moreover, at the smaller scales there are two main perspectives within the approach with a focus on the linear network representation: (1) strokes; and (2) segment generalization. The stroke-based generalization groups the road segments into longer lines, which may cross without explicitly intersecting. The decision to intersect may be based on some criteria, such as geometry (angle between segments), topology (node degree of two), attribute or classification (Zhou and Li, 2012; Thomson and Richardson, 1999). In the segment-based generalization, the road segments (from junction to junction, where the topological degree of end nodes is greater than two) are the smallest atomic elements for removal.

The main advantage of using strokes is evident: it preserves information about the connectivity between segments. This indicates that the stroke-based approach can be a useful generalization tool. However, Turner (2007) points out that segment analysis (the creation of a segment map as known by the space syntax community) can give comparable or better output compared to a stroke-based method and even produces more meaningful results, *e. g.* better in correlation with observed vehicular flow.

From computational perspective, the strokes approach is used for all road segments present in the dataset, but for our purpose only local information is needed, *e. g.* to find the road segment with best continuity to determine the new classification. Additionally, our road network is changing dynamically during the process and it would be useless to maintain global information about strokes. Therefore, we only view a merged set of road segments as a stroke to determine its new classification, when there is no 'clearer' alternative to do this.

There has also been continuous map generalization research concerning road network, see Section 2.3. Another truly continuous solution is proposed by Li and Zhou (2012), including experiments with real data and an extensive evaluation. They compare both approaches; strokes and mesh density, and combine them in their so-called *integrated* method to create a universal solution for road network using the advantages of both approaches. Their solution generates two separate linear and areal hierarchies, combining them together to provide continuous multi-scale representations of a road network. This rather complex solution is based on the omission of features, with the analysis of what should be eliminated. This analysis takes place only at the beginning of the process. Changing parameters of the generalization process are not considered; *e. g.* rules/parameters adapted during different phases of the generalization. On the other hand, the performance test looks promising and suggests that the approach is quite feasible and will be good for on-the-fly use.

§ 4.3 Concepts and definitions

A processing strategy for generalization throughout the scales is introduced in Section 4.4. Before doing so, terminology and specific tools are explained in this section to be able to better explain our strategy. Note that Section 4.3.1 and Section 4.3.2 explain theoretical concepts first. Then Section 4.3.3, 4.3.4 and 4.3.5 will present more practical tools implemented in our developing kit.

§ 4.3.1 Granularity

Continuous generalization requires geometric changes (big or small) between generalization steps in the process. We call the number of features (data) changing in one generalization step the *granularity* (Šuba et al., 2015), and we distinguish the following levels:

- *course granularity*, when all features are involved at once, *e. g.* all roads are omitted.
- *medium granularity*, when all features of a certain class or subclass are processed, *e. g.* all local roads with speed limits are removed.
- *fine granularity* when one single feature is processed, *e. g.* one dead-end road is removed.
- *finest granularity*, when a part of a single feature is involved, *e. g.* one road segment is removed.

Since we aim at a more gradual transition without significant modification in geometry, the changes should be small. In our case, *finest granularity* is then optimal. This guarantees that the changes are as small as possible, aligning well with the vario-scale concept.

We apply merge/remove, collapse/split and simplification operations for these object parts, which can result in a road object composed from segments represented by mixed lines and faces; see Figure 4.2. From a traditional cartographic point of view this might seem less favorable, but from a vario-scale point of view, it is desirable. The differences in the representation can be compensated in the visualization by applying proper styling, when the line segment is represented by lines of the same thickness equal to the width of the adjacent area segment. Furthermore, the fact that the generalization process is performed in small steps leads to simpler problems, which are easier to compute or implement. On top of that, the history of steps is stored explicitly, and this implies that the links between generalized and original objects are present. Often, these links are missing in multi-scale implementations.

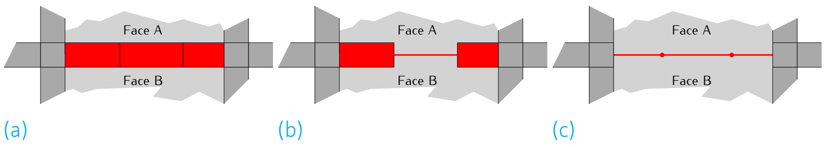


FIGURE 4.2 Side effect of the gradual transition from one scale (a) to another (c). For some reasons (perhaps different attributes; *e. g.* road surface type, name or speed limit), the red road consists of three parts. To achieve a gradual transition, the individual parts are generalized separately. It changes the representation from areal at the most detailed scale (a) to semi-linear at the 'halfway' scale (b) to linear at the final scale (c). Be aware of the fact that the complete road in one moment of the process is represented by both areal and linear parts at the same time. Note that one of the consequences is a topological change, where *Face A* and *Face B* become adjacent.

§ 4.3.2 Level of abstraction

The input datasets that are currently supported within the tGAP structure are modelled as a two-dimensional polygonal map, *i. e.* as a partition of the plane in a geometric sense, without gaps and overlaps. As a result, the data structure contains only topological primitives; nodes, edges and faces, where one area object in the map corresponds to just one topological face. The same is true for roads in the input dataset, these are represented by faces only (areas).

Besides classification and the planar area partition, there is more semantic information *implicitly* present in the large-scale input map, such as the linear networks (road infrastructure or rivers and water channels). These linear networks are implicit in the input data, and we wish to preserve their natural meaning in the target map of the smallest scale as well. Even thought these features are part of a network they are often not explicitly modelled. Therefore, we will first make the implicit information about the role features play inside the network explicit, even when not given as input.

Figure 4.3 shows a simple example of such a network in the input (large scale) (see Figure 4.3a) and in the target small scale (see Figure 4.3d). The road network can be easily derived from both figures. When two road segments are both represented by areas (large scale), then they are incident when they share an edge. Where at least one of the road segments is not represented by an area, then they are incident when they share at least a point (node). However, it is not so simple to keep track of the linear network as the map changes from scale to scale in a more gradual way; see the intermediate steps in Figure 4.3b, c. Therefore, the same situation is captured in Figure 4.4, but this time, the *linear graph* of the road network is depicted for better understanding of the relationships among the road objects. As mentioned, the road segments can be *incident* with other road segments. Depending on the number of incidences, a road segment plays the role of junction (node) or connection (edge) in the linear network graph. From Figure 4.3a–d, we can see that the geometric embedding of the objects in the map changes, while the conceptual linear network graph stays the same. This gives us an effective tool for meaningful road network generalization throughout the scales. Figure 4.5 shows actual structure to represent this phenomen.

Keep in mind that in the following description, the road segments of different dimensions can be in an interaction with each other, *e. g.* a 1D line can be in an interaction with a 2D area road segment. Therefore, the road segments can be classified (especially in gradual changing scales) based on the number of other incident road segments and are included in the data structure as follows:

- A road segment is classified as an *isolated segment* when it has no other road segments incident.
- A road segment is classified as a *dead end* if it has exactly one other road segment incident. It is represented either by a face or by an edge in the topological data structure.
- A road segment is classified as *road connection* of the network when it is incident with exactly two other road segments. It is represented either by a face or by an edge in the data structure.
- A road segment is classified as *road junction* of the network when incident to more than two other road segments. It is represented either by a face or by a node in the data structure.

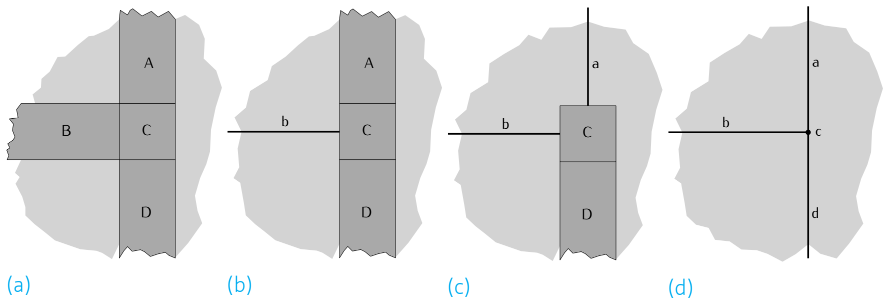


FIGURE 4.3 Example of road network generalization from the large scale (a), through intermediate step (b, c), to the final scale (d). Geometrical representation of the road segment/junction changes from 2D areas to 1D segments/0D point, but 'features' (semantic), and their role in linear network remain the same.

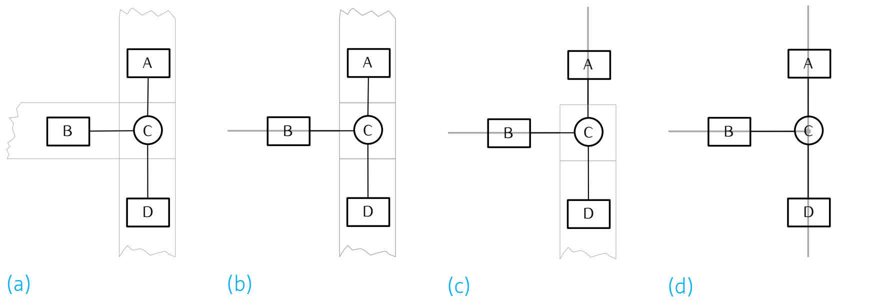
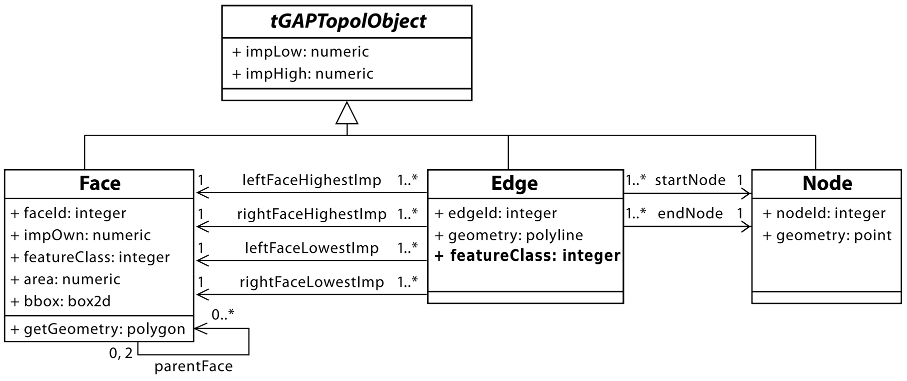


FIGURE 4.4 The linear network graph represents the same situation as Figure 4.3. It captures the topological relationships of road objects where a rectangle indicates a road connection and a circle a road junction.

In this way, we can associate any selected feature in the map at any stage of the process with a topological feature. The assumption related to the input data is that connections and junctions of a road network are well defined in the input dataset; see Figure 4.6a. However, when the input data do not conform to this assumption (see for an example



(a)UML diagram, the modifications to capture road networks are in bold

```
CREATE TABLE tgap_faces (
face_id integer,
imp_low numeric,
imp_high numeric,
imp_own numeric,
feature_class_id integer,
area numeric,
bbox box2d);
```

(b)Face table

```
CREATE TABLE tgap_face_hierarchy (
face_id integer,
parent_face_id integer,
imp_low numeric,
imp_high numeric);
```

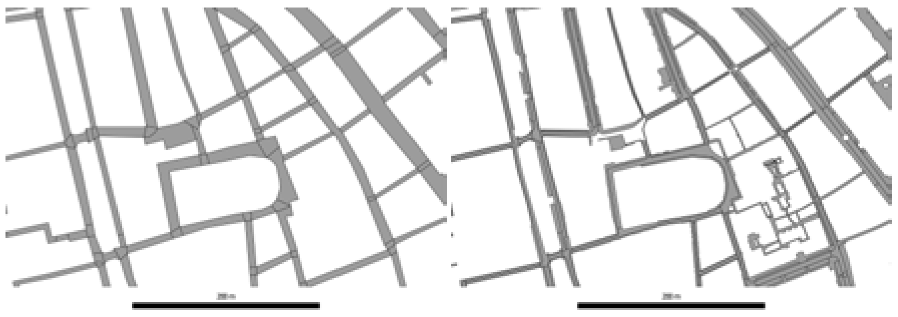
(c)Face hierarchy table

```
CREATE TABLE tgap_edges (
edge_id integer,
imp_low numeric,
imp_high numeric,
start_node_id integer,
end_node_id integer,
left_face_lowest_imp integer,
right_face_lowest_imp integer,
left_face_highest_imp integer,
right_face_highest_imp integer,
feature_class_id integer, #feature class for collapsed features
geometry geometry);
```

(d)Edge table

FIGURE 4.5 Road representations in UML diagram and the database tables for tGAP (It is inspired by (Meijers, 2011, p. 159)).

Figure 4.6b), an additional pre-processing step must be taken. This can be done by applying the constrained Delaunay triangulation to obtain properly-classified (connection or junction) area road segments, as proposed by Uitermark et al. (1999).



(a) (b)

FIGURE 4.6 Two map fragments of different possible input showing the city center of Leiden, the Netherlands: first (a), the topographic map (TOP10NL) intended for use at a 1:10,000 map scale; second (b), the BGT base map (in Dutch: Basisregistratie Grootschalige Topografie) intended for use at a 1:500–1:5,000 map scale. Note that only road network features are displayed.

§ 4.3.3 I-neighbours

Operations in our process are performed locally based on the global criterion of least important feature. To identify how many objects are involved in an operation we introduce $i - neighbours$. This is similar to a breadth-first search, *e. g.* $1 - neighbours$ checks only the direct neighbours of the chosen object, $2 - neighbours$ checks the neighbours of the direct neighbours, $3 - neighbours$ checks also the neighbours of the $2 - neighbours$, etc.

This topological measure can be used in two ways either for faces or for edges. Figure 4.7 shows the principals for each representation.

§ 4.3.4 Connectivity

When collapsed features (lines) interact, *e. g.* the classification for a newly-merged road must be picked, we define how well they fit together by computing the $connectivity$ value. For a specific road segment it is defined by how many routes between other segments go via this segment. More passing routes within segments means higher $connectivity$. This measure is known from space-syntax theory (Turner, 2007), which is used in analysis for urban planning and implemented in software, such as depthmapX[1]. There exists a modified version of the connectivity value, called *radius connectivity*. Here a geometric criterion ϵ, where $\epsilon \in \mathbb{R}$, is used. For the chosen segment only the

1 `https://varoudis.github.io/depthmapX/`

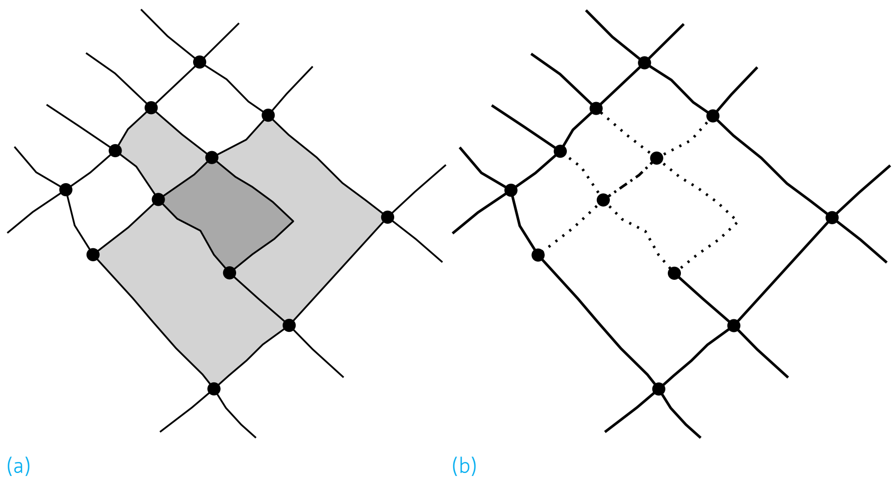


FIGURE 4.7 $1-neighbours$ for a face (a), where the selected face is in dark grey. In (b), $1-neighbours$ (dotted) for one selected edge (dash-dotted).

other segments within a radius ϵ of the segment are considered. It seems to hide the fact that segments on the edge of dataset have by definition a lower $connectivity$ value than segments in the middle of the dataset.

In our case, we compute the $connectivity$ locally. But instead of using a geometric radius ϵ, we use a topological measure, only considering $i-neighbours$, see Section 4.3.3. Figure 4.8 illustrates the computation of the $connectivity$. Let's assume that we want to identify the connectivity for selected road segment (in red). In this example, red segment has $connectivity = 6$ for the $1-neighbours$ and $connectivity = 16$ for the $2-neighbours$. Note that in our implementation other aspects such as classification or length of segments are involved.

§ 4.3.5 Deflection angle

To find the best continuing neighbouring road segment (for lines), we use a simple geometric principle, that is called the deflection angle. Figure 4.9a illustrates that this is the deviation from $180\,°$ of the angle between two road segments. This is inspired by the strokes approach (note that many strategies for stroke building exist, cf. Zhou and Li (2012)). By using the deflection angle, we form locally a 'stroke' where a neighbouring segment appears to follow the same direction as the merged segment if the deflection angle is small. However, if the deflection angle is large (*e. g.* more $90\,°$) the road segments will not be going in the same direction. We chose a $60\,°$ threshold, if an adjacent, neighbouring segment has a deflection angle larger than this, it will not be considered.

Figure 4.9 demonstrates the principles of determining the new classification based on the deflection angle. Let's assume that the decision has been made to merge newly collapsed roads e_2 and e_3, see Figure 4.9b. There are three classification possible for the new segment e_{23}; 'A','B' and 'C', where 'A' is the highest. The segment e_1 has the

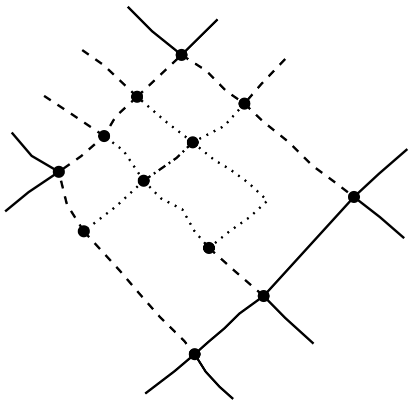
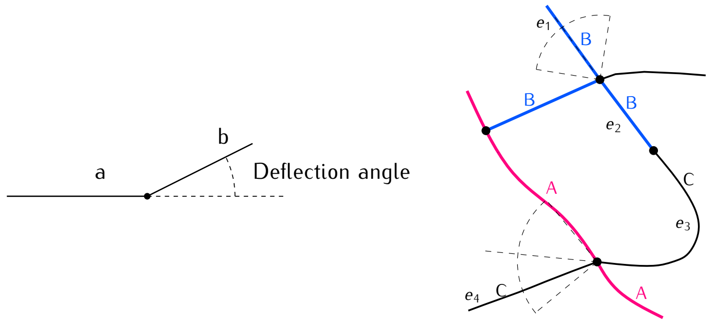


FIGURE 4.8 An example of defining connectivity for one selected road segment where only collapsed features (roads) exist. The dash-dot segment is selected. $1 - neighbours$ are dotted. $2 - neighbours$ are dashed.



(a)The principle of deflection angle. A smaller angle means a better connectivity.

(b)An example of merging two road segment. The deflection angle is indicated with the dashed lines.

FIGURE 4.9 Determining the classification of a merged segment, where the original segments are classified differently

best deflection angle of all adjacent segments and its classification is used to determine the class of the newly merged segment. Note that the classification e_1 only indicates what the new classification should be and it thus not used directly.

One of the three possibilities can happen; class of $e_1 \geq$ maximum of classes e_2, e_3, class of $e_1 \leq$ minimum of classes e_2, e_3, or class of e_1 is between classes e_2, e_3. In the first case, the final class will be the maximum of the classes e_2, e_3. In the second case, the final class will be the minimum of classes e_2, e_3 and in the third case, the class will be based on the other aspects such as the *length* of segments e_2, e_3. In example 4.9b, the new merged segment will carry 'B', because $e_1[B] \geq e_2[B]$.

§ 4.4 Strategy for complete scale range of road network

This section introduces our proposed processing strategy for road network generalization. The generalization process to generate content for the vario-scale data structure is based on the tGAP-principle (find the least important object and merge it with a most compatible neighbor) extended using linear network knowledge. Note that the principle is designed in a very generic way, such that it is possible to mix different generalization operations. It is also integrated in the sense that all features for which the operations are performed are treated together (all features are geometrically integrated in the same planar area partition that is used as input). There are many design decisions in the development of this process, and we will label these as *DESIGN DECISION I* (with *I* a sequence number). Often, there are several alternative options, but based on our experience and some limited testing, we present our initial 'best guesses' for these decisions.

Design Decision

Design Decision 1: We distinguish in the creation of the vario-scale content for the data structure three classes of objects only: roads (sub-classified as either junction or connection), water and other objects. This will reduce the number of object types during the creation of the vario-scale to three, which makes decisions more transparent. Besides road network processing, this also allows us to treat water differently from other non-road classes. Note that the original classification of the other classes is kept and used later on in visualization (but not during the creation of the vario-scale structure). Alternative processing design decisions here could have been made: two classes (road, other: even simpler), three classes (roads, water and other: more refined, with two subclasses for water: junction, connection), more classes (as present on common topographic base maps, where the classification of these other classes is used; *e. g.* for selecting the most compatible neighbour to merge with).

At the beginning of the process, every face in the structure gets an importance value based on the type and the size (area) of the feature (in the initial large-scale map, there are only area features as a constraint). Note that the computation of the importance value can be refined; see Section 4.7. Based on the importance value for every face, the process starts picking one face after another and performs specific actions based on the type of the chosen face. The face with the lowest importance value is processed first.

Design Decision

Design Decision 2: As in the integrated data structure, both area and line representations of roads (and other features) are possible; an alternative to having just faces in the importance queue is also having line or node features in the importance queue.

Depending on the type of a face, there are the following processing options; see Figure 4.10.

- The selected face is a *road junction* and will be either merged with the adjacent road junction or preserved until all adjacent road connections are collapsed. If the former, then the face itself can be collapsed. If the latter, then the importance is raised and the face is put back in the queue (and will be processed later). Note that this can cause infinite loop, therefore an additional measures are taken such as queue reordering.

  *Design Decision 3:*Instead of postponing the processing of the junction, it would be possible to directly collapse it to a node (even if not all incident connections are collapsed).

  Design Decision

- The selected face is a *road connection* and will be merged with the adjacent road connection. If there is no such face, then it is collapsed to a line.
- The selected face is *water* and will be merged with another adjacent water face. If there is no such face, then it is collapsed to a line.
- The selected face is the *other* object and will be merged with an adjacent other object, if there is any; otherwise, the face is collapsed.

  Design Decision 4: If no adjacent face of type *other* is present, an alternative design decision instead of collapse would be to raise the importance and put it back in the queue. Later on, when one (or more) of the neighbour road faces have been collapsed to a line, then the selected face might have an adjacent *other* face.

  Design Decision

  Adjacent *other* objects with no collapsed roads lying between are the most optimal to merge with. If there is no such adjacent option, another object with the least important collapsed road (edge) between is selected. When the collapsed road lies between, the faces will be merged (and the collapsed road will be removed).

This recipe guarantees to generalize the roads in a meaningful way and is continuous for all faces in the structure. Roughly speaking the following happens with the roads: Initially, the area road segments are collapsed, and later on, the merging of the other areas takes place. To which neighbour the other area is merged depends on the edge between. If there is no collapsed (line) road segment in between, this has preference. If all edges represent collapsed road segments, then the least important one is selected, and this decides with which neighbour area it should be merged. With this type of merge, the unimportant linear road segments are 'automatically' removed, as well.

It is important that the least important collapsed road (line) is determined by looking at its classification, the local configuration (connectivity) and length. Specifically, for every potentially collapsed road, we look at its classification first. If this gives a 'winner', we pick this road as the least important collapsed road. If this results in a tie (collapsed roads having the same classification), we compute for every collapsed road a connectivity value. The connectivity value for one road is defined by how many routes between other roads go via this road, with more passing routes meaning higher connectivity, see Section 4.3.4. The least important collapsed road is the one with the lowest connectivity value. As a last resort, in the case of roads also having the same connectivity

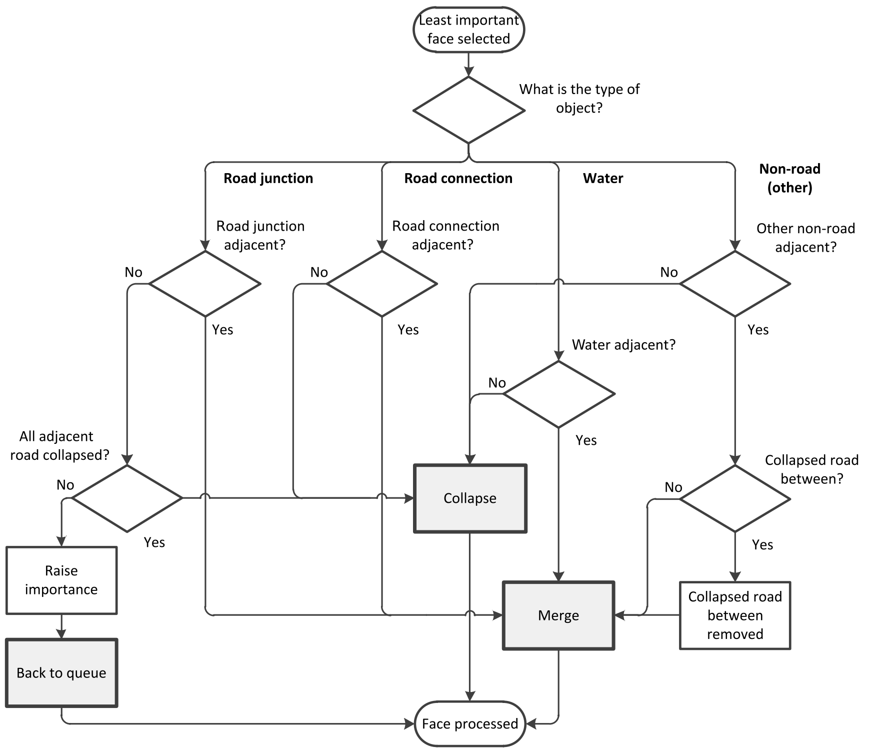


FIGURE 4.10 The way one generalization step is performed. Note that the road area will never merge with a non-road (other) area.

value, the shortest of the collapsed roads is selected as the least important one. An in-depth description is given in (Šuba et al., 2014b), where this has been extensively tested.

Note that another (non-road) area is collapsed if and only if it is completely surrounded by areas of roads, *e. g.* a face of grass is between two faces of road (grass strip between lanes of a highway). This is a rare case where the collapse of a non-road feature is the most favourable. Another option would be to return the non-road face back into the queue and wait until at least one road nearby is collapsed (this is *DESIGN DECISION 4*). However, the collapse operation is preferred in this case, because it makes sense to assign the parts of this unimportant face to the neighboring road faces, and it will reduce the number of faces by one.

DESIGN DECISION

DESIGN DECISION 5: In the beginning of this section, we defined road junctions and road connections. During the generalization process the configuration changes, which gives two options: (1) faces keep their original road (junction/connection) subclassification even if it is in contradiction with our definitions; or (2) faces are reclassified when needed in order to remain consistent with our definitions for junctions and connections. We opt for the first option, because this results in a slightly better vario-scale cartographic quality according to our visual inspection; *cf.* Section 4.5.3.

The above-described iterative generalization proceeds in steps. Normally, there is one face fewer after every step, and the number of faces never increases. There is some delay when a road junction area still has a road connection area as a neighbour, causing the processing of the junction to be postponed. However, the neighbouring road connection areas will all be collapsed to lines at a certain moment in the process, and after that, the delayed processing of the road junction areas can take place.

§ 4.5 Experiences with a data set with terrains and integrated roads

This section addresses the cartographic quality (Section 4.5.1), the quantitative analysis carried out (Section 4.5.2) and additional discussion items (Section 4.5.3).

§ 4.5.1 Cartographic quality

For our experiments, we loaded a subset of the Dutch topographic map (TOP10NL) intended for use at 1:10,000. Two following regions were used:

- A rural region, area of 7×7 km with 11,300 faces as input; see Figures 4.11 and 4.12.
- A city center (of the city of Leiden). It is a 1×1 km region with 19,400 faces as the input; see Figure 4.13.

These datasets are provided as simple feature polygons, where terrain, water and road network layers together form a complete planar area partition. The road segments are present in the dataset with geometries for road junctions and connections. The layer of buildings lies on top of those layers. First, we 'fused' all data layers together and created a planar partition as input that we converted to a topological data structure with the help of software FME[2]. At this stage, all objects are represented by areal geometries. Then, this input is processed into a vario-scale structure with the help of merge/remove and split/collapse (Meijers et al., 2016) generalization operations. Line simplification could be included in the process; see Figure 4.14c, d. However, this is not the case at this stage of the research process in order to see the features' geometry without any additional effects.

Figure 4.11 shows the impact of our method on real data. It demonstrates that by making small generalization steps, we got incrementally a simpler map. At intermediate scales two representations for road objects (areas and lines) are mixed.

Figure 4.12 presents a sequence of maps for a small part of the rural region retrieved from the tGAP structure as generated by our algorithm. It demonstrates the outcome of the algorithm (vario-scale content), but it does not correspond to the correct user impression as the scale is fixed in this figure and vario-scale map use should be an interactive experience during zooming. At least it gives an impression of the content of the different map scales. Figure 4.13 shows a proportional re-sized map sequence to give a better impression of how a user should perceive the derived maps.

2 `http://www.safe.com/fme/`

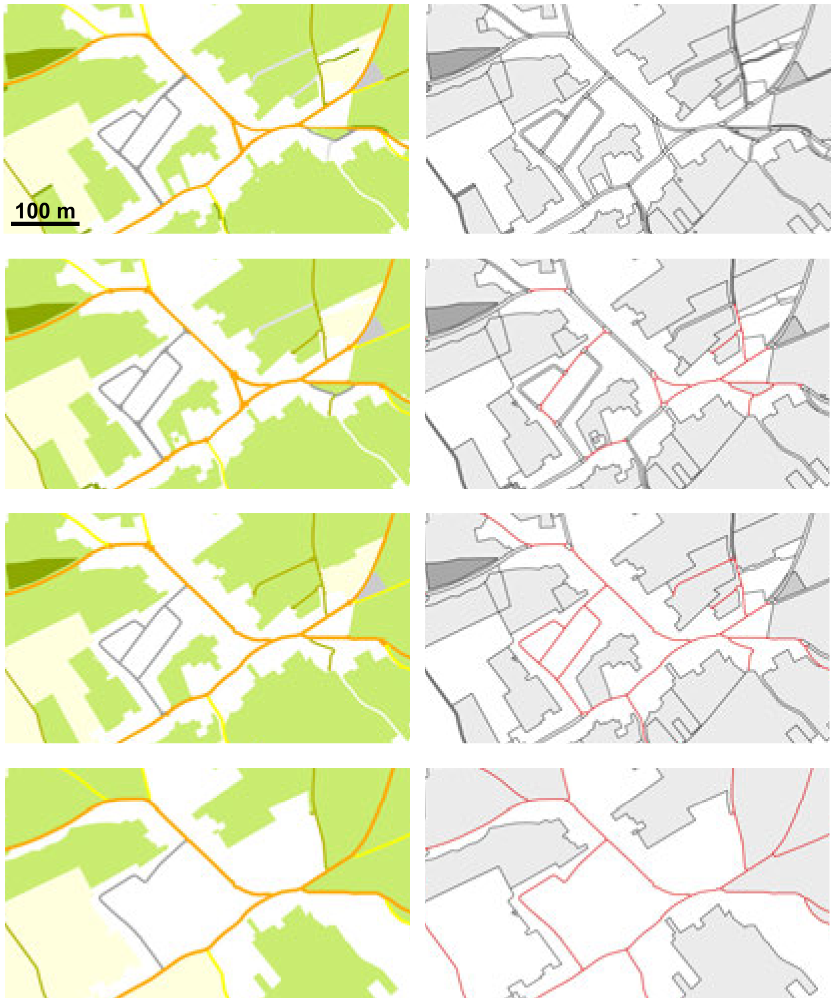


FIGURE 4.11 Detailed situation throughout the scales. All maps are at the same scale (and the exact scale is not very relevant). It illustrates how the structure evolves. The top maps are at the input scale, where all objects are represented by areas. The intermediate scales are in the middle, where the representation of roads is mixed (by areas and by lines) and generalized scales are at the bottom, where roads are represented only by lines. The styled map (left) corresponds to the situation in the tGAP structure (right). Red lines (right) indicate collapsed road edges. Note: all map fragments are displayed at the same scale to clearly show the effect of the generalization process (in reality, the bottom map fragments should be shown at smaller scales).

Figure 4.14 presents a detailed situation of a simple roundabout. Note that these types of infrastructure objects are not present as separate entities, nor classified as such; their road segments are dealt with individually.

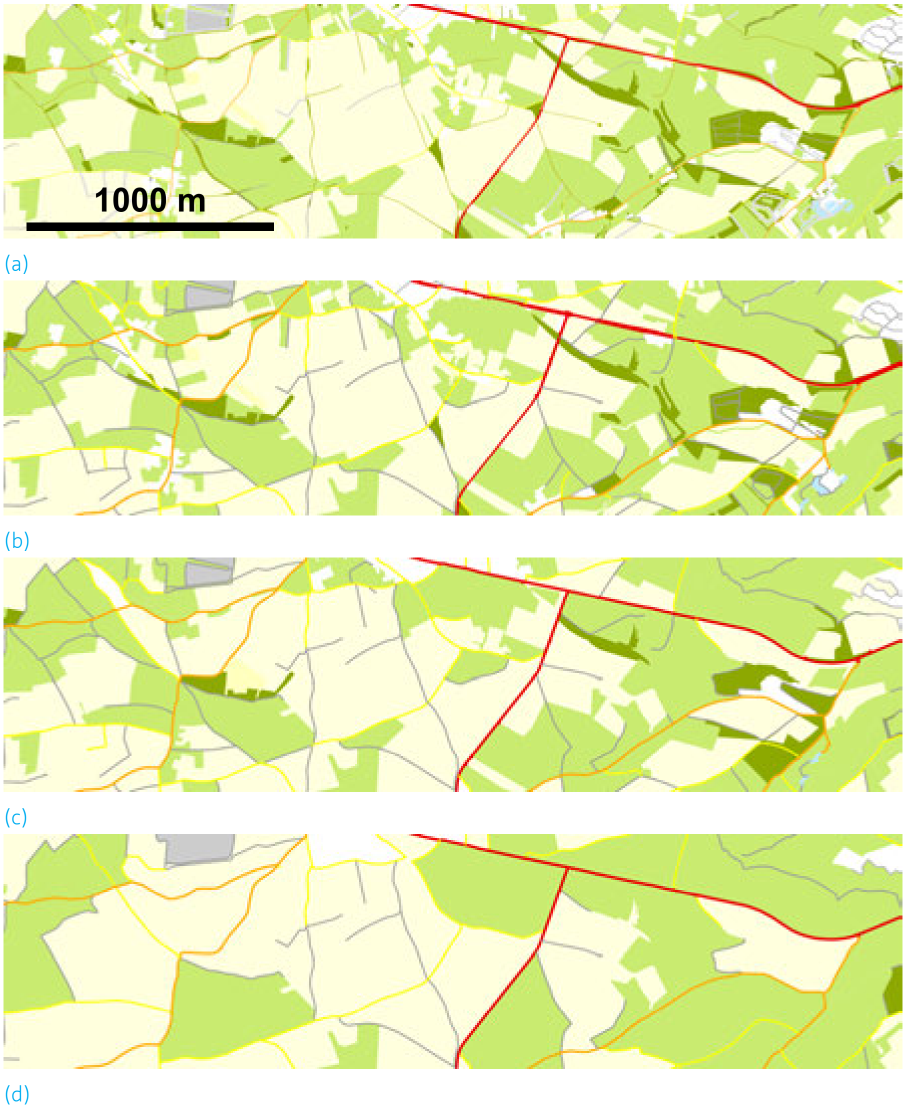
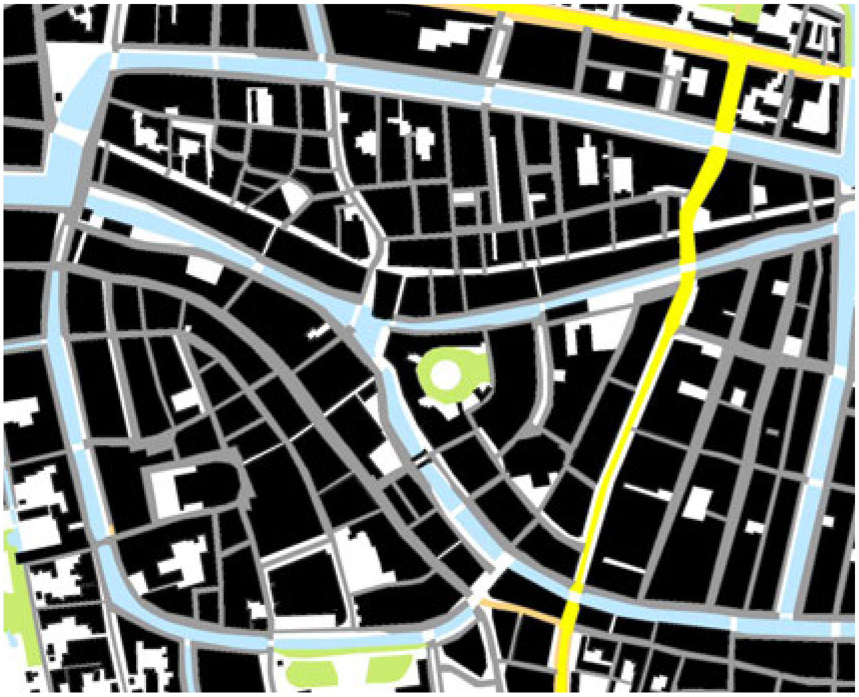
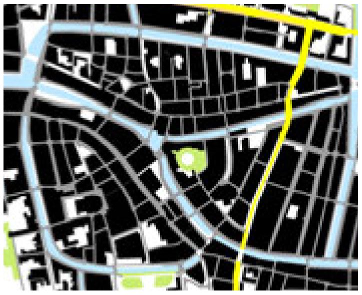
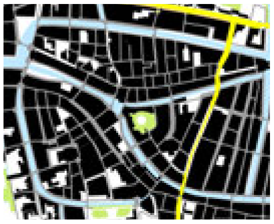
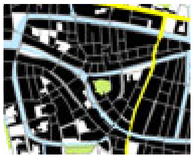


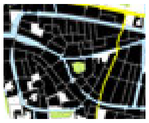
FIGURE 4.12 An example of the generalization process in the rural region (presented at the same scale). The top figure (a) shows the input. The other figures (b), (c) and (d) show map fragments after 86.9%, 94.4% and 98.2% of the process, where 0% not generalised (a), with 11,300 faces and 100% is reduced to one face.


(a)


(b)


(c)


(d)


(e)

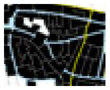

(f)

FIGURE 4.13 An example of the generalization process for the city center dataset. (a) shows the input; The other subfigures (b–f) represent 60.6%, 68.7%, 80.8%, 88.9% and 95.0% of the process, where 0% is the input with 19,400 faces and 100% is reduced to one face. Note: the map fragments are displayed at the intended target scales; (a) 1:10 k; (b) 1:16 k; (c) 1:18 k; (d) 1:23 k; (e) 1:30 k; (f) 1:45 k.

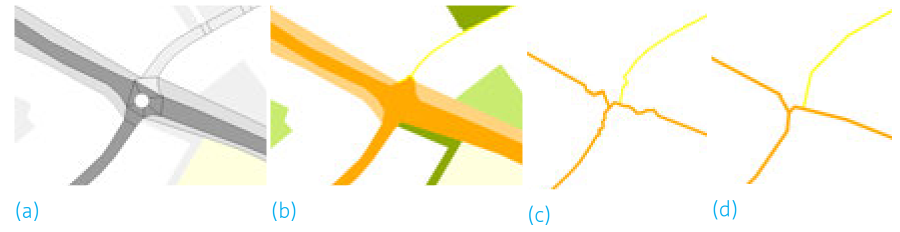


FIGURE 4.14 Example of the process for a roundabout. It changes the representation from areal in the initial partition in (a), partial areal and partial linear in (b) to linear (c). (d) demonstrates the option when line simplification is used.

§ 4.5.2 Quantitative analysis

Due to the nature of illustrations on paper, only specific intermediate map scales can be shown. The intention is to use the vario-scale data in combination with interactive zooming and panning operations over the map. However, there are only a few existing measures to evaluate continuous map generalization in general. Thus, we used visual inspection and compared our results to previous developments. To give a better quantitative notion of the process, we also generated some graphs, which give better insights into the whole generalization process.

First, Figure 4.15 shows the proportion of feature classes throughout the generalization process. For every generalization step, we count how many objects in the tGAP structure there are for a certain feature class. Then, those feature classes are grouped into 'water', 'terrain', 'buildings', 'roads' and 'other' super classes. This graph corresponds to the example depicted in Figure 4.12. Similarly, Figure 4.16 shows the area these objects cover in the structure. Note that a collapsed road has no area (even if it is still a map object) and that measured in the area of the roads has a smaller share than when expressed proportionally (%). Near the end of the graph in Figure 4.16, road objects do not occupy any area, although there are still road line segments. The water bodies are small and occupy only a small portion of dataset; therefore, they do not survive long in the process.

Second, Figures 4.17 and 4.18 provide another indication of the same example from Figure 4.12. With proper styling and color schema, it is not obvious which roads are still areas and which have been already collapsed. Therefore, these graphs present absolute numbers of road objects. Figure 4.17 captures only roads objects represented by areas. Figure 4.18 shows the number of edges in the structure (representing the collapsed road objects). It shows that the process collapses the majority of road areas first (creating new road lines). Later in the process – part *b*, the road lines between two merged faces are removed. This corresponds to our designed strategy.

Graphs presented so far have shown only small value changes. It indicates a gradual process that corresponds to our goal presented earlier. It also suggests that our generalization rules in the overall strategy were quite reasonable.

Finally, Figure 4.19 shows the usage of the different generalization operators throughout the process. Exactly one operator is applied in every generalization step, either the merge/remove or the split/collapse. The graph summarizes what happened ev-

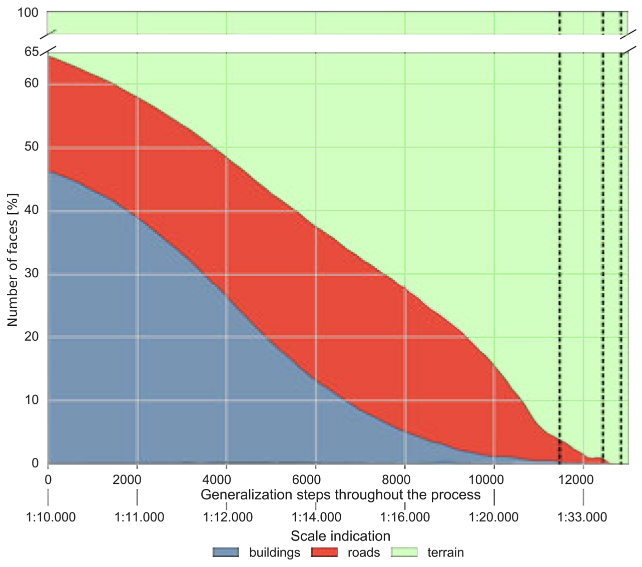


FIGURE 4.15 Ratio of feature classes throughout the generalization process in topological Generalized Area Partition (tGAP). The numbers are relative (number of faces per feature class divided by all faces present in the map). Note that the total number of objects decrease throughout the process. The vertical dashed lines indicate the map scales shown in Figure 4.12b–d.

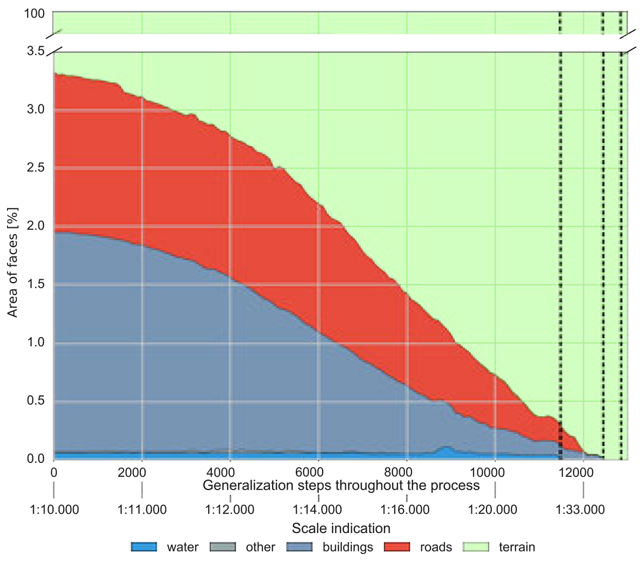


FIGURE 4.16 Ratio between the covered area in tGAP structure for feature classes throughout the process. The graph relates to the data of Figure 4.12. The vertical dashed lines indicate the map scale of Figure 4.12b–d. The numbers are relative.

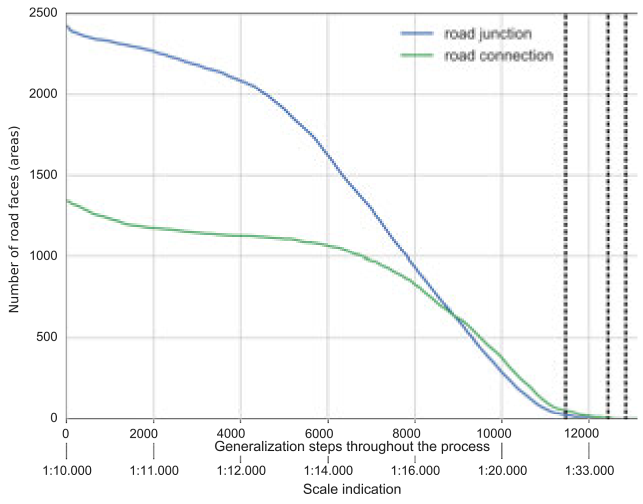


FIGURE 4.17 Number of road faces represented by areas throughout the process in Figure 4.12.

ery 500 steps and shows the ratio between the operators. One can observe that a lot of merge/remove operations happen at the beginning when tiny faces are merged. Those faces are mainly slivers from the preprocessing step when layers of buildings were 'fused' together with other layers. Since they are small, they have low importance and are processed first, but this could have been an additional preprocessing/cleaning step.

Later in the process, the split/collapse operator is more dominant because road objects are processed. Finally, the merge operator becomes more significant again because most of the roads are collapsed, and other objects are then merged together, removing any collapsed roads between.

§ 4.5.3 Additional discussion points

The results above have shown some reasonably good outcomes in automated and continuous map generalization. However, there are still quite a number of design decisions (as mentioned above) and some additional issues, which have been encountered during the design and implementation. In most cases, the best solution is not yet known, and further research is needed. The list of additional issues includes:

ROAD CLASSIFICATION Road objects were classified as *road junction* or *road connection*, based on the number of incident road segments, where a *junction* should have more than two road neighbours. There are two options when this classification can take place (*Design Decision 5*). In the first option, the objects are classified in a preprocessing step, and then the same knowledge is used throughout the whole process. In the second option – more dynamic, the objects are reclassified during the process, when needed.

Figure 4.20 shows the processing sequence for both approaches, starting from an initial configuration where humans would recognize the strip of grass between two roads running in parallel. We can see that static classification (on top in the figure) identifies two junctions and two connections at the beginning of the process. Then, connections

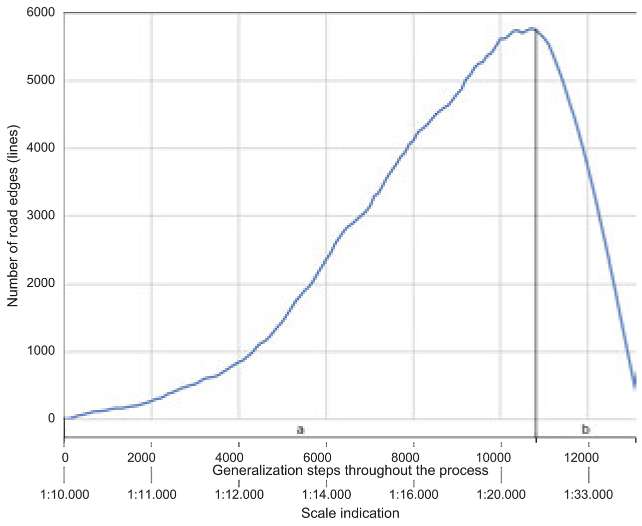


FIGURE 4.18 Number of road line edges in the structure (collapsed roads), related to Figure 4.12. Graph part *a*, where the number of collapsed road connections increases, represents the initial stage of the process, where roads are split/collapsed. Part *b*, where the graph decreases, indicates a situation where two area objects merge, intentionally removing the collapsed road lines between.

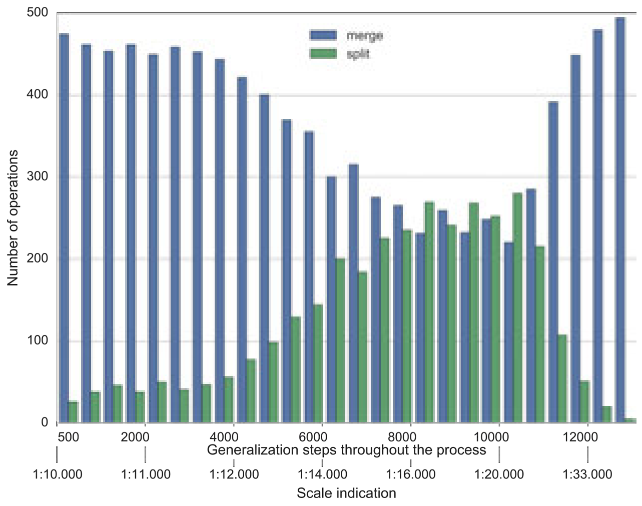


FIGURE 4.19 Application of the merge/remove and the split/collapse generalization operator throughout the generalization process. It is related to the example in Figure 4.12.

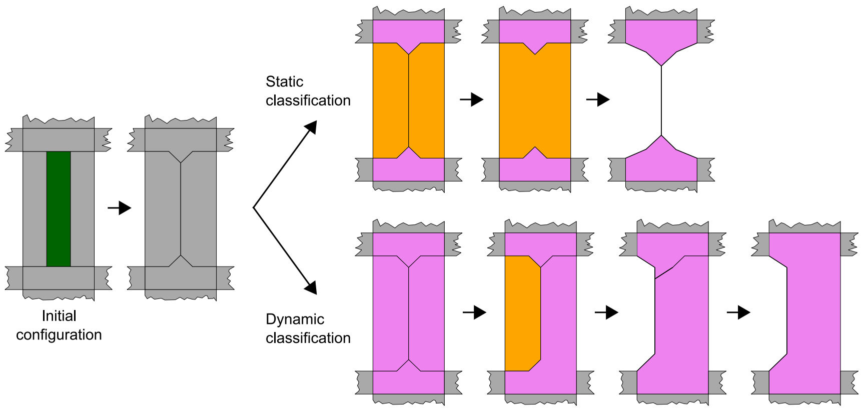


FIGURE 4.20 Alternate road object classification during the generalization process. It starts with collapsing the strip of grass between roads. Road junctions in pink and road connections in orange.

are merged together and then collapsed. On the other hand, dynamic classification (at the bottom) recognizes everything as junction objects (all objects have more than two incident roads). Then, two road junctions are merged, and a new classification identifies one new road connection, which is collapsed later. The last step is the merge of the two remaining road junctions.

The generalization process would continue and remaining junction(s) would be collapsed. Note that for dynamic classification this would happen much later because the final junction is larger (higher importance) and would lead to a different geometry of collapsed roads.

Besides technical aspects, such as memory use or time complexity, a static approach may lead to incorrect classification (of junctions and connections according to our definition) during the process, but it gives a slightly better overall cartographic impression during visualization. Therefore, we used it in our implementation.

RECLASSIFICATION AFTER SPLIT/COLLAPSE *Design Decision 6:* When the face of the road object is collapsed, the newly-created edges should carry a correct classification; see Figure 4.21. The face is transformed based on the skeletonization to a set of edges. Most of the time, one 'main' branch corresponds to the shape of the original face. However, the most appropriate reclassification for new edges is not so clear. Should all branches receive the same classification or should only the main branch be classified?

Design Decision

New classification for all edges guarantees good connectivity, because it is more easily detected in the topology. The implementation is simpler and more obvious for further processing. Therefore, we used it in our approach. Nevertheless, it slightly deforms the original network. The network is 'spreading'; see the middle in Figure 4.21 and the detail in Figure 4.22. On the other hand, another option is to classify only the main branch of the collapsed object. This way, the road network is prevented from unwanted spreading. However, the whole network in the domain becomes more and more shattered throughout the process and connectivity analysis will be more difficult.

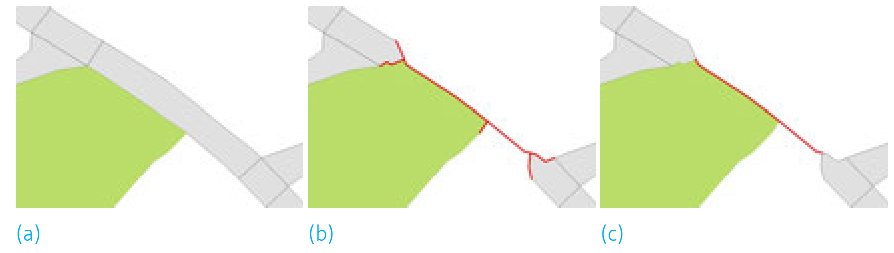


FIGURE 4.21 Reclassification after split/collapsed operation. One road connection object from the initial configuration (a) can be reclassified in two ways. Either all branches (in red) carry road classification (b) or only the main branch carries information (c).

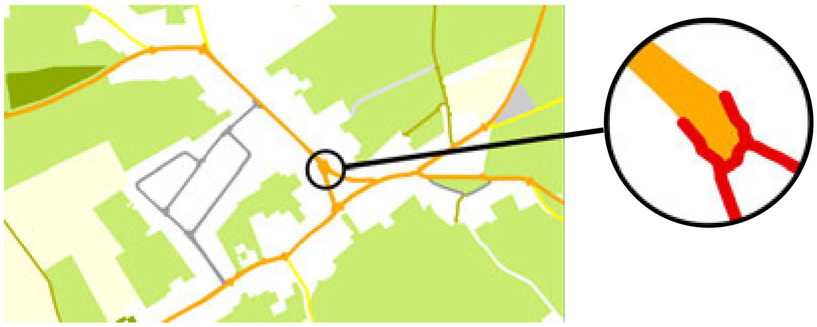


FIGURE 4.22 Side effect of all branches reclassification. This detail relates to Figure 4.11.

CROSSING OF MULTIPLE NETWORKS The input map is a projection of 3D space into a 2D map. This means that the information about linear networks crossing each other at multiple height levels should be preserved somehow in the map, allowing us to use that knowledge in the processing. However, what should we do if there is no so such knowledge available? Obtaining data is one thing (*e. g.* if it is available in TOP10NL), but how do we keep that knowledge during the process when objects change their representation? Additionally, how should the priority of individual networks be set? Another not really clear aspect is the solution of the case where more important networks cross less important ones, *e. g.* a road network crossing rivers and water channels. What to do in such a case is not really clear at this moment.

It is interesting to note a possible gain in the connectivity of two linear networks: roads and water. At the larger scale and with a single classification per face, it is not possible to model the fact that both the road network and the river network are properly connected. However, at the smaller scales, when the road and water segments are collapsed to lines and nodes, both networks stay connected at these locations (and better represent the nature of both networks).

§ 4.6 Better classification with groups

The current approach lacks a solution where the map features such as groups of buildings forming built-up area should stay longer in the generalization process rather than falling in the shattered pieces as seen in Figure 4.23. Here, we will presented better strategy for aggregating features.

The creation of tGAP structure is driven be two main aspects; First, the global order of the features based on importance value, *i. e.* in every step the least importance feature is selected and it is processed alone (collapsed) or interacted with adjacent features. Second, the most compatible neighbour(s). Our assumption based on visual inspections is that the first has 20% influence in the appearance of the final result compare to 80% for the second. Therefore, the second aspect is more relevant and has been researched before. Other previously developed strategies for vario-scale process can be summarized as following:

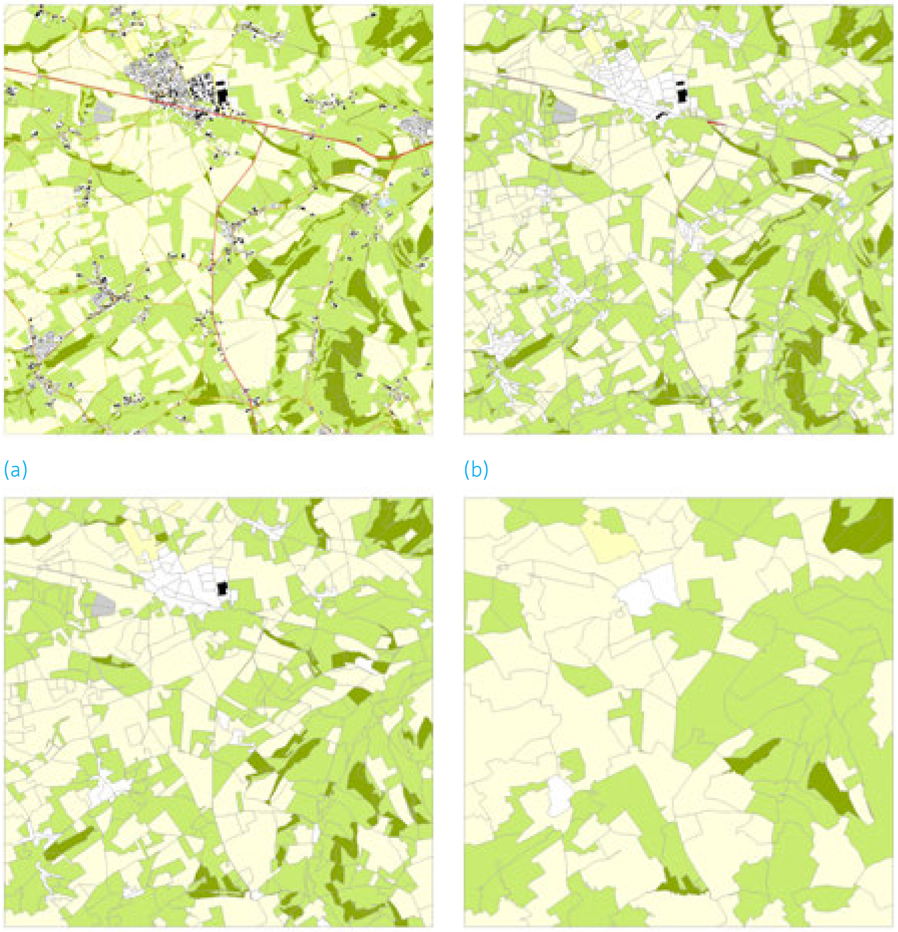
COMPATIBILITY MATRIX van Putten and van Oosterom (1998) provided full description of principle and it is captured in Section 3.2. The principle is based on most compatible feature class which is used for picking the most compatible neighbour. All possible combination are captured in the predefined compatibility matrix.

On one hand, the main advantage is the simplicity of the solution where same set of rules captured in the matrix stay the same throughout the process where the matrix defines relations between the feature classes which is based on analysing of existing data (and collect statistic about objects) . There is also a classification hierarchy in the matrix by definition. On the other hand, it suffers from the fact that the matrix for the whole process is designed based on educated guess and trial and error.

Figure 4.23 demonstrates the result of generalization with simple a compatibility matrix. Note that resulting map 'falls apart' into pieces quite early during the generalization process. The small objects are eliminated, other faces are merged into objects of similar sizes and patterns such as streets, villages or grouped settlements are hard to distinguish, giving the user an artificial impression. This problem is even more obvious later, see Figure 4.23c and d.

CONSTRAINED TGAP Haunert et al. (2009); Dilo et al. (2009) proposed an other method of capturing better map content in the generalization process, see Section 3.3. Beside the source scale it uses a final generalized map (small scale). This final small scale map is obtained either from other sources or computed in advance. Then it generates all intermediate scales between in such a way that the final sequence of maps progresses between the large and small scale maps. The most significant limitation of using constrained tGAP is in the requirement to use additional knowledge derived from other solution. In theory, it means that the solution cannot run independently. Moreover, the results are very dependent on the 'target' small scale approach obtained from complex and expensive preprocessing step or other source.

Besides the mentioned approaches above there are other researchers who tackled similar problems in an automated generalization process. Haunert and Wolff (2006) used a very expensive aggregation function to identify the possible groups of features which should be handled and preserved throughout the process. Zhang (2012) used a method based on geometric configuration only, *i. e.* clusters of building or build-up


(a)

(b)

(c)

(d)

FIGURE 4.23 Generalization process where compatibility matrix been used. (a) shows the input. The other figures (b), (c) and (d) show map fragments after 87 %, 93% and 98% of the process, where 0% is the input (a) with 11,300 faces and 100% is one face.

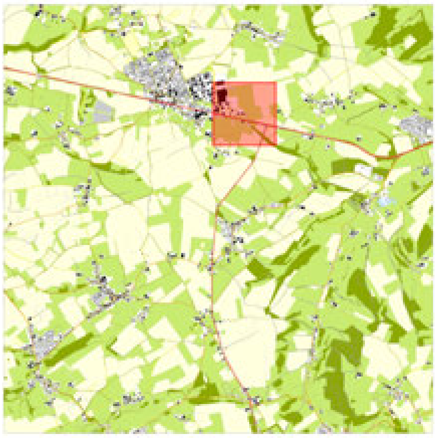


FIGURE 4.24 The same test dataset as in Figure 4.23. An example of groups definition as shown in Figure 4.25 took place in the rectangle region.

areas. It uses Delaunay triangulation and Minimum Spanning Trees structures to construct a proximity graph defining the building clusters.

Even with those proposed strategies we want something more generic. We seek a very generic solution which can; 1) be easily implemented and used for arbitrary input data, 2) discourage certain merges to happen while features are potentially quite compatible, and 3) be obtained from multiple sources such as classification, geometry, or other. Therefore we propose a new method for *the most compatible neighbour* selection. The innovative idea is based on the object's membership in predefined sets and we call the method *groups*. It uses the fact that objects are members of the same groups to influence compatibility value. Note that the individual groups can be overlapping and one object can be member of the multiple groups. More practically, for the selected object we compare all adjacent objects and the one with the highest number of common groups is chosen as *the most compatible neighbour*. Other parts of the generalization process stays the same and the generalization actions are performed the same way.

At the beginning of the process the groups have to be defined. They can be obtained from attributes of the input data and/or as a result of the geometrical test. Selection of all local roads can be the example of the first, faces completely fitting a polygon representing build-up area can be the second. Obtaining the groups from attributes is straightforward, however obtaining the groups based on geometrical test requires a layer of geometry to test against. There are more options how this layer can be obtained. The considered options follow (together with indication of our experiences within brackets):

- A layer of geometry is obtained from existing maps of target scale (not tested). Then the map objects are tested against the layer *i. e.* faces completely fitting the polygons will be members of the group. This option has same limitation as the constrained tGAP method; (1) the quality of result is dependent on additional source material and (2) the fact that it cannot run independently.

- Triangulation of the building geometries layer first and then elimination of triangulation edges crossing the roads (not tested). The group membership is defined based on remaining clusters of objects still connected by triangulation edges. The method is inspired by similar strategy for the displacement operation Zhang (2012).
- A layer of geometry is created by buffer of the building geometries (tested). It is inspired by research conducted by geo-ICT Geodan[3].

Figure 4.25 shows an example of the groups generated in a test dataset. See the overview in Figure 4.24, where four groups have been created. First three g_1, g_2 and g_3 are retrieved from the *feature class* attribute of the input. g_1 with the smallest regions is defined by all roads, see Figure 4.25a. g_2 with bigger regions is defined only by local and the most importance roads, see Figure 4.25b. Large blocks in g_3 are defined by the main roads (highway) and hydrology, see Figure 4.25d. The urban regions in $g4$ are defined by geometrical test against a combinations of buffers around the buildings; We used buffers of the building geometries (to grow) and the buffers of negative value (to shrink). The combination of buffers is used to create a geometry representing build-up area where buildings are close to each other and details and holes in geometry are removed .

In the phase when the selection of the most compatible neighbour take place it uses the following formula for every possible neighbour:

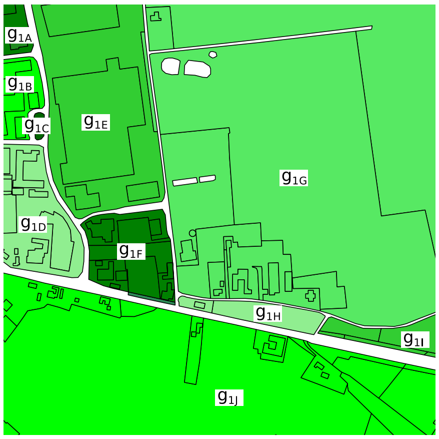
$$compatibility_{f,n} = \frac{|G_{common}|}{|G_{total}|} + l^{-10}$$

where $compatibility$ defines a value between the least important face f and neighbour n. $|G_{common}|$ specifies the number of common groups of f and n and $|G_{total}|$ shows total number of the groups in dataset. l is the length of the common boundary between f and n. The value l plays minor role in breaking a tie between two possible neighbours with the same $compatibility$. The potential neighbour with the higher compatibility will be chosen for the most compatible neighbour. Note that for a newly merged face a membership in the groups must be defined. We used attribute intersection of previous two memberships, no new testing is performed.

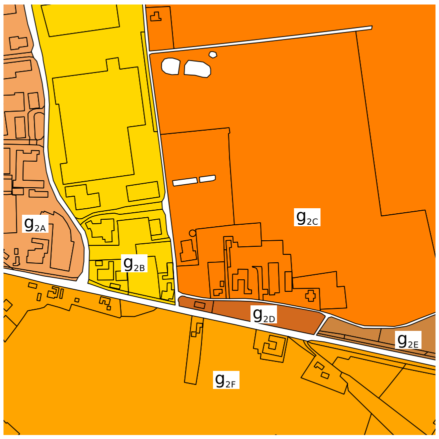
Figure 4.26 shows complete generalization process where groups were applied. To make it more transparent only the merge operation is used. A good indication can be the dominant highway in the top half of the figure running from west to east. It is preserved throughout the entire process, which was our initial intention. We can observe that road features are present much longer in the process, instead of approx. 60% in the process captured by Figure 4.23. On the other hand the roads features become more dominant in later stage (depicted by the yellow and orange regions in Figure 4.26d). Note that this would probably not happen when split operation is used as well.

We can observe two interesting things; The first, the faces are merged within the same groups if possible. The second, since the size of the features drives the whole generalization process it has a side effect on the groups. The groups with smaller regions

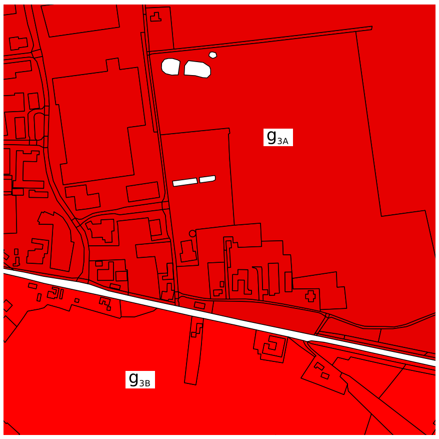
3 http://research.geodan.nl/sites/bag-tiles/



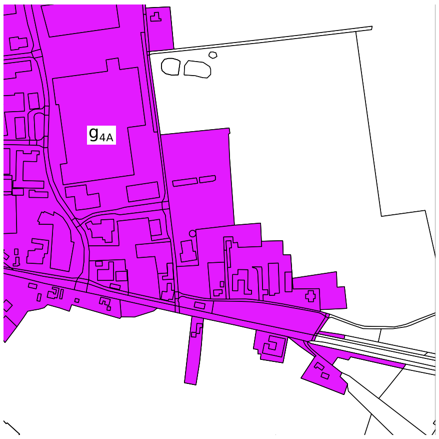
(a) Green polygons are members of g_1. The group is defined by all roads.



(b) The objects in shades of orange are in the group g_2. Provincial roads, highways and hydrology define the group.

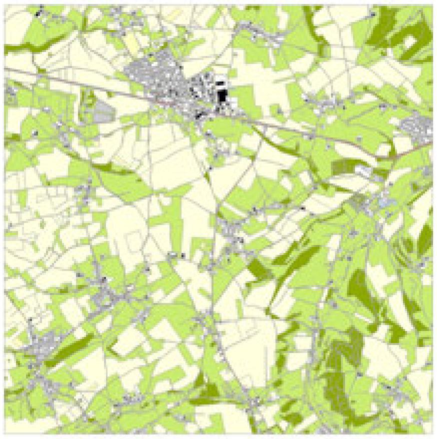


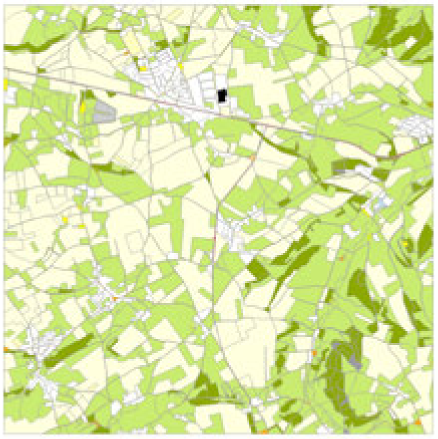
(c) The objects in red are in the group g_3. Hydrology and highway define the group.

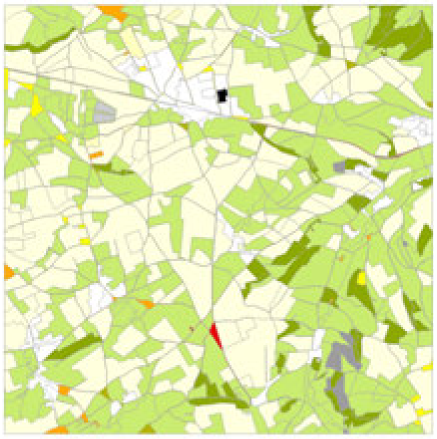


(d) The objects (in purple) define urban regions of g_4.

FIGURE 4.25 Four defined groups g_1, g_2, g_3 and g_4.


(a)


(b)


(c)

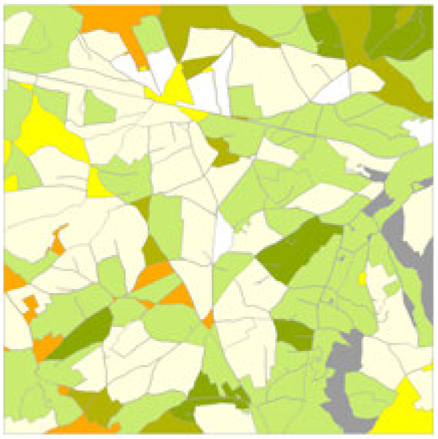

(d)

FIGURE 4.26 Generalization process where only merge operation with groups have been used. The figure (a) shows the input. The other figures (b), (c) and (d) show map fragments after 87 %, 93% and 98% of the process, where 0% is input (a) with 11,300 faces and 100% is one face.

are processed first followed by the groups with bigger areas, and so on. This puts the groups in implicit hierarchical order. In our case g_1 is dissolved first and one group covering all of the dataset remains at the end.

Another advantage comes from the fact that additional information is obtained from the same input. Moreover, it is not dependent on the setting of a precise threshold, only checking object membership. This makes the group method very generic, cheap and effective, and worthy of future investigation. In this section, we have presented the group method only to show the principle. However, we assume better results can be obtain in combination with the compatibility matrix, where the matrix steers the process at local level (within small building blocks) and the group method at global level (urban - rural regions).

§ 4.7 Conclusions

Up to now, our vario-scale method could only be used to represent area features. We have shown that line features in our current vario-scale solution may be introduced. In Section 4.4 we have designed an algorithm which provides the fully-automated generalization process that preserves a road network throughout all scales, and which gives reasonable results. The algorithm maintains knowledge about a road network even in situations where roads are partially represented by lines, as well as partially represented by areas. We presented the necessary modifications that have to be applied to accommodate such an algorithm and still follow the idea of small generalization steps. For this, we have introduced new concepts, techniques and more processing strategies in Section 4.3 and presented six design decisions. We also have suggested that it is sufficient to use a large-scale planar partition with only area objects and their classification as input. We have validated our approach on a test dataset together with some quantitative measurements. With the generated results, we now have an opportunity to conduct user testing of the vario-scale principle. However, there are still the following issues that need to be researched:

- Other options of the various design decisions as mentioned in this paper could be further analysed.
- The advanced treatment for water networks (also junctions and connections), rail networks, buildings or other feature types could be included. It might be very well the case that during the generalization process additional knowledge or different treatment is needed for these types.
- Line simplification in the generalization process could be included to create the vario-scale data structure. However, standard line simplification techniques could introduce extraneous line intersections. Therefore more care needs to be taken with non-trivial simplification, considering topological correctness of results.
- Besides `Top10NL` data, the proposed approach should also be applied to other datasets; *e. g.* Corine (smaller starting scale) or Dutch BGT (Basisregistratie Grootschalige Topografie) (larger starting scale).
- The proposed strategy using groups requires more testing. Neither the optimum number of groups nor limitation of the size of the dataset is known, *i. e.* can we obtain same result for 200 step process as for 2,000 step process?

5 Large vario-scale datasets

In Chapter 3 the focus was on vario-scale data structure description. This was extended in Chapter 4, where generating better content for this structure was investigated. It showed how the structure has been developed and used in practice, and current technical limitations. One of them is processing really massive dataset with records in order of millions which do not fit in the main memory of computer. It is a notorious and challenging problem. This is especially true in the case of map generalization, where the relationships between (adjacent) features in the map must be considered. Therefore, this chapter presents our solution for automated generalization in vario-scale structure based on the idea of subdividing the workload according to a multi-level structure of the space, allowing parallel processing. More specifically: Section 5.1 specifies our goal. Section 5.2 presents related work and other options to handle large datasets. Section 5.3 explains the principles of our method in more detail. In Section 5.4 modifications of the process specific for road network generalization are introduced. Statistics and a test of real dataset with more than 800 thousand objects are given in Section 5.5, followed by conclusions and the future work related to processing large datasets in Section 5.6

Own publications

This chapter is based on the following of my own publications:

- Meijers, M., Šuba, R., and van Oosterom, P. (2015). Parallel creation of vario-scale data structures for large datasets. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-4/W7:1–9.

§ 5.1 Requirements

Geographic vector datasets are an example of the big data phenomenon. Practitioners using these large geographic datasets of the whole world, a continent or a country, for example, can easily get into difficulties because the sheer size of the data is too big. Functionalities such as storage, analysing, processing or visualisation hit the physical limitations of a computer when treating all data at once. Automated map generalization is not exceptional in this. This is also true for the process of generating vario-scale data structures, see Chapters 3 and 4. Only datasets which can fit into the main memory of the computer can be processed efficiently, limiting the maximum size of datasets to be processed. Therefore, we have developed an automated generalization method for processing large geographic vector dataset of arbitrary size into vario-scale data structures.

The main goal of our research for this chapter is to create a process to be able to produce a vario-scale data structure for a large dataset based on the following requirements (each one supported by its own motivation):

- Input independent – The approach should be able to process different source datasets intended for various map scales.
- Automatically generated partition – The division of the space into smaller pieces should work fully automatically and not require manual intervention.
- Suitable for parallel processing – Besides the fact that datasets hitting the physical limitation of the computer cannot be processed, the approach should be suitable to run in parallel for more practical reasons, such as reduction of processing time.
- The features should be processed just once – Features on the boundaries of a dataset split in pieces will require special care. However, it should be guaranteed that features should not be split in artificial parts, and the feature is processed just once during the whole process, increasing the efficiency and consistency of the result.
- Be suitable for data produced at a range of map scales – A vario-scale structure contains a whole series of simpler maps where features in the map are modified versions of their own predecessor. This requires a hierarchical data structure to record all the successive changes.

The described method in this chapter fulfils the requirements listed above. It fits to one of our goals to produce a vario-scale topographic dataset (for example based on the large scale 1:10,000 map for the whole of The Netherlands) and disseminate this data via a web service interface enabling access at arbitrary map scale and smooth zoom user interactions over the web, see Section 3.6. This dataset's planar area partition (roads, water, terrain) contains approximately 5.2 million area features. To be able to process a massive dataset of this size we have used the concept of a Fieldtree (Frank and Barrera, 1990) which was proposed quite some time ago to be applied to create tGAP (van Putten and van Oosterom, 1998), but has never been implemented nor tested. The method splits the original dataset into smaller pieces, called *fields* (at multiple levels). Starting at the lowest level, these small fields can be handled separately, and processed fields (with less data after generalization) are combined into bigger fields at a higher level. An additional benefit is the possibility to process the fields in parallel on today's multi-core systems, which can significantly speed up the generalization process. The objects located on the boundaries of the fields require special attention, which will be closely described later in the text.

§ 5.2 Other approaches

Automated map generalization of large datasets is a notoriously difficult problem: It is a computationally complex, time demanding and data-intensive problem and it requires high-performance computing, commonly considered as one main approach for handling such data. Using cluster-, grid-, cloud- or super computing, parallel processing is a good way to deal with massive datasets (Sharker and Karimi, 2014). It requires decomposition into independent tasks that can run in parallel. This can be executed in less time, leading to more efficient computation and solving of complex problems, which map generalization unarguably is. These reasons and the fact that computers with multiple cores are common in these days explains why researchers focus on map generalization of large datasets in parallel.

Many aspects of processing geographic data in parallel have been extensively studied in the past. Domain decomposition for parallel processing of spatial problems has been studied by Ding and Densham (1996), Armstrong and Densham (1992), Zhou et al. (1998) and Meng et al. (2007). Meng et al. (2007), for instance, used the Hilbert space filling curve to achieve a better parallel partitioning. Another focus point has been on the development of frameworks for parallel computing (see *e.g.* Hawick et al. (2003), Wang et al. (2011) and Guan et al. (2012)). The MapReduce programming model (Dean and Ghemawat, 2004) is well known for parallel computation and has only recently been extended with capabilities for various computational geometry problems (Eldawy et al., 2013). It must be noted, that all these works do not focus on the problem of automated map generalization.

Meijers and Ledoux (2013) present an algorithm, termed EdgeCrack, to obtain a topological data structure and perform small geometric corrections of the input by snapping to avoid problems. They show how they extend the algorithm to a Divide-and-Conquer approach using a quadtree based subdivision, which is also suited for parallel processing. They report on successfully obtaining a topological data structure for 5.3 million polygons with their approach, however it does not consider any generalization.

The Netherlands' Kadaster processes the whole of The Netherlands from scale 1:10,000 to 1:50,000 in an automated fashion in parallel (Stoter et al., 2014). They developed their custom solution using ESRI generalization software. To obtain a map for the whole of the country they partitioned the dataset based on the main road network. Their irregular partitioning requires minimal splitting of features and the amount of data is more less distributed equally. This advantage comes with drawbacks, because defining this partitions is difficult and cannot be done fully automatically (*e.g.* near the coast the road network is too sparse to obtain reasonable small dataset parts). Their solution is created and tuned for generalization of maps for a specific target scale and to achieve the best result possible, however this tailor-made solution can not be applied out of the box for different dataset or targeting creation of data for another map scale such as a European land use map.

Thiemann et al. (2011) present the automated derivation of CORINE Land Cover (scale 1:100,000) from the high resolution German authoritative land cover datasets (scale 1:10,000) of the whole area of Germany. First the datasets are split into rectangular and slightly overlapping tiles. These are processed independently and then composed into one result. To preserve consistency of data some redundancy is added to the partitions in the form of overlapping border regions. In these border regions features are processed more than once, leading to redundant (and potentially inconsistent) output. This redundancy is removed in the composition phase. Their generalization uses some deterministic algorithms for every tile to guarantee that the tiles can be stitched together without any observed problems later. A user parameter that describes the size of the overlapping border regions is required.

```
NRLEVELS=8       # Given number of levels to create
FINEGRID=12500   # Grid size (m) at most detailed level
THEORIGX=0       # Origin in x- and y-direction
THEORIGY=0

for (i=0; i<NRLEVELS; i++) do
    SIZE[i] = FINEGRID * 2^i

ORIGX[NRLEVELS-1] = THEORIGX
ORIGY[NRLEVELS-1] = THEORIGY
for (i = NRLEVELS-2; i>=0; i--) do
    ORIGX[i] = ORIGX[i+1] - SIZE[i+1]/4
    ORIGY[i] = ORIGY[i+1] - SIZE[i+1]/4
```

FIGURE 5.1 Pseudo code as given by van Oosterom and Vijlbrief (1996) for determining the lay-out of a Fieldtree (its grid size and the shifted origin of the grid cells at every level). Note that the number of levels (needed to be able to fit the whole domain of the dataset in one last field) can be determined given the parameter for the finest grid size (finegrid) and the extent of the dataset.

§ 5.3 Generating a large vario-scale structure

The whole process of obtaining a large vario-scale data structure is composed of a sequence of steps: Data import, constructing a Fieldtree and distribution of the objects over the Fieldtree, processing field by field (level by level) and finalization. The description of the complete process will now follow in the same order in sections, step by step.

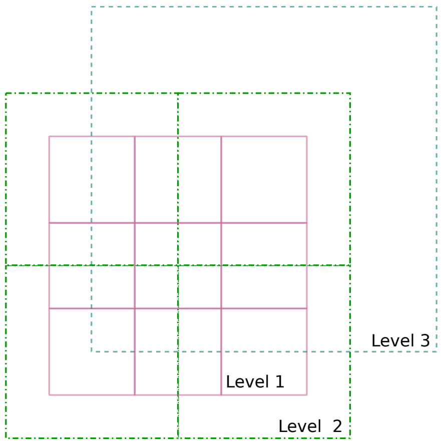
§ 5.3.1 Import

The initial step of the process is to import the planar partition into a topological data structure consisting of nodes, edges and faces. We consider topological clean data where nodes with a degree of two do not exist (unless for island rings) and there are no intersecting or dangling edges. If the input data is not in the required topological structure (as data is often modelled as a set of simple features) and the dataset is large, then this is also a large data processing challenge, to which the same parallel processing strategy could have been applied as presented in this paper.

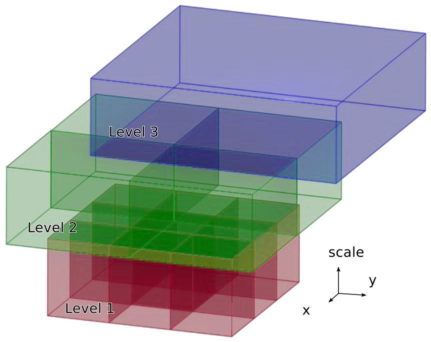
§ 5.3.2 Constructing a Fieldtree and object distribution

To split the dataset into smaller pieces we use the Fieldtree data structure (Frank and Barrera, 1990). The Fieldtree has been designed for GIS and similar applications. Besides its multi-level organization of space, it subdivides space regularly, spatial objects are never fragmented, and geometric and semantic information can be used to assign the location of a particular object in the Fieldtree.

Figure 5.2 demonstrates the organization of fields at multiple levels in the Fieldtree. The most detailed and smallest fields are at the bottom (large scale), and the less detailed and larger fields (small scale) are at the top of the Fieldtree. The bottom fields will initially be filled with the input data, while the content of the higher level fields is



(a)Three levels where lowest is in pink, middle in green and top level in blue.



(b)3D impression of the same fields. Note, the third dimension is scale.

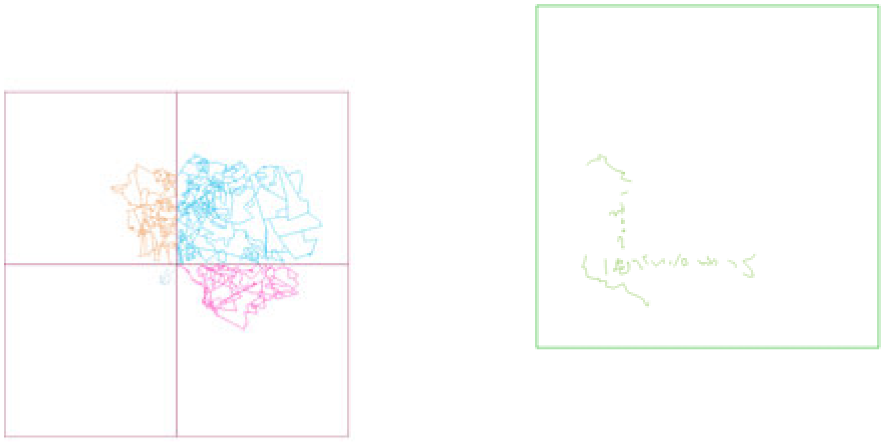
FIGURE 5.2 A Fieldtree (top view and 3D impression)

(mainly) generated during the generalization process. Note that due to the shifted grid origins, every field has nine child fields (of which 1 completely contained, 4 half contained, and 4 quarter contained) in the level below, except the fields from the lowest level. In the reverse direction: a field has either 1, 2 or 4 parents (depending on its location in the grid). The fact that a field can have more than one parent causes it to be a non-hierarchy and strictly speaking the term 'tree' is incorrect. The number of levels in the Fieldtree is based on the extent of the dataset, and the size of the lowest level field is given by a user defined parameter (*cf.* Figure 5.1).

When the layout of the fields for all levels in the tree has been determined the features of the dataset are redistributed over the fields. The nodes, edges and faces are distributed to the fields in the lowest level based on their bounding box, see Figure 5.3. Every feature is stored in the smallest field in which it is completely contained. If the object is bigger than a field or it intersects with more fields, a non-fitting object is placed in a field at a higher level and it will be processed later (together with the generalized features of the child fields). It has been proven that an object is completely contained in a field not more than two levels higher, compared to its own size. A very large object will be placed in one of the highest levels of the Fieldtree, which means that such an object will not be processed (generalized) until that level is reached.

Once the distribution is finished, the lowest level fields can be treated separately in parallel. Every field has it own set of tables with nodes, edges and faces in the database: When a field is processed these tables provide the initial input; during processing these are populated with the result of the generalization steps and at the end of processing output is written to the appropriate parent field tables.

Note that only features completely present in a field are processed at the current level (*i. e.* for an area feature (face) all its composing nodes and edges are present in the field). Features on the boundary of a field are processed at a higher level, where they completely fit. When all the nine child fields are processed, then the parent field one



(a)Lowest level fields (b)Highest field

FIGURE 5.3 Distribution of the edges over the fields of the Fieldtree. Data that fits in a field has its own colour. Edges that intersect the boundary of one of the fields at the lowest level, have been stored in the field at the next level. Note the shift in origin of the field at the highest level.

lever higher can be processed. This processing of fields is repeated until the top level is completed.

The grid location at the next level is shifted, in order to guarantee that objects which could not be processed in the current field (*e. g.* they are bigger than the field) can be treated in the next or one level above that. Note that every field of the next level has edges of twice the size (and therefore four times the area size) compared to the fields at the current level. It is our goal to simplify the content of a field by a factor 4, that is, if the input consists of n area features, then the output should be of size $n/4$. In a uniform distribution this will result in higher level fields with similar workload as in the lower level fields. In case of non-uniform data the reduction by a factor of 4 will maintain the relative differences in map density, while decreasing the number of features towards the higher (and larger) level fields.

§ 5.3.3 Processing Field by Field

When the features have been distributed over the fields of the Fieldtree every field in the lowest level can be processed. Processing fields (*i. e.* executing the automated generalization process for this part of the map and keeping track of the result in a tGAP structure) is the most time and computational consuming part. But because objects in the different fields are not in interaction with each other, every field can be processed separately and this processing can happen in parallel. Note that our automated generalization process is based on iteratively finding the least important feature in the current field, instead of a global criterion, this has now become a 'local field' criterion. This makes sense as generalization of for example the northern part of the dataset, should not influence much the generalization that happens in the south. Furthermore, replacing the global by a local criterion makes the problem computationally easier.

Processing one field means applying generalization operations: Merging faces, removing boundaries (edges), simplification of edges, collapsing areas to line features and splitting of faces. This leads to simplification of the objects of the part of the map stored in the field and divides generalized features into two categories. First, finished features: these are not valid any more for the map scale that has been reached with the generalization process. These features are stored for the final vario-scale dataset. Second, unfinished features: these still need to be processed (together with the 'boundary' features, which did not fit in the lower level fields and the features that were intersecting the field boundaries of the just processed level). These features are placed in the tables of the grid cell of the next level in the Fieldtree in which they completely fit. We call this *propagating* features up. As explained above, to preserve the same amount of information (processing work) through the whole Fieldtree about 75% of the field is processed. The other 25% is left and propagated up. Because the area of a field at the next level is four times bigger, the data amount (and thus the workload) for a field at the next level will be approximately the same.

§ 5.3.4 Finalization

After all fields at all levels of the Fieldtree are processed the final operation takes place; it combines all information together. All processed fields are searched and the final global ordering (and numbering) of the objects based on their range of scales defined during the vario-scale structure creation takes place. This global ordering is helpful when using the structure for making maps at arbitrary map scale. The individual field tables are merged together to create one set of tables (with one node, edge and face table for the final dataset) that encodes the result of the progressive generalization process for the whole dataset. At this stage the structure is ready to use by other applications such as web clients supporting smooth zoom of vario-scale data.

§ 5.4 Read-only buffer zone for road network

When we tried to use our Fieldtree approach for a road network dataset of The Netherlands (in Dutch: Nationaal Wegen Bestand), we realised that for generalization nearby features are important (*e. g.* in the connectivity analysis needed in road network generalization), and this was slightly problematic with our design where features overlapping tile boundaries are already propagated up (not present at lower levels).

We outlined in Section 5.3 that objects on the boundary of a field can not use information about their direct neighbours for generalization decisions, because they are not present (having already been propagated up). However, in case of processing the road network, we need extra information at the boundary of the field because the connectivity of the roads is crucial, because roads incident with many other roads should stay in the process the longest. Therefore we introduced a 'read-only' data buffer around the boundary of the field where no generalization actions can be performed and objects are used only for connectivity information purpose, see the red features in Figure 5.4. The buffer is computed locally. Instead of using a geometric radius ϵ, we use a topological measure, only considering i-neighbours, see Section 4.3.3. It is similar to a breadth-first search, *e. g.* 1-neighbours check only direct neighbours, 2-neighbours check the neighbours of the direct neighbours, etc. In our case, first the faces intersecting with

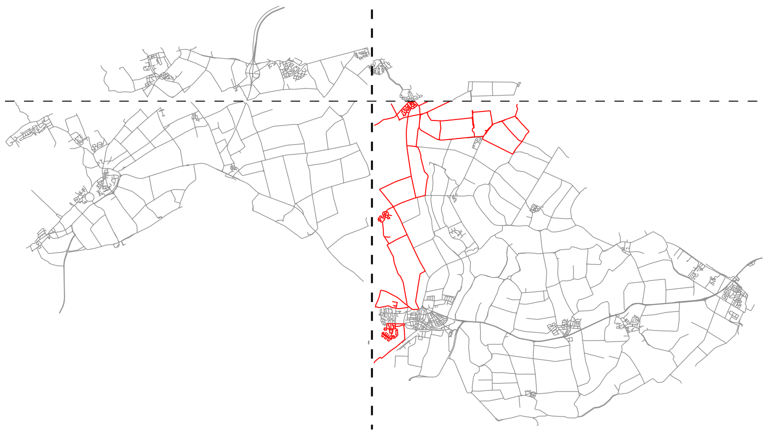


FIGURE 5.4 The inward 'read-only' buffer example for a subset of the road network dataset with only four field at the bottom. The buffer for right bottom field in the figure is illustrated in red. $2 - neighbours$ of the intersecting face with boundary of field has been chosen. Note two things: First, only the content of the fields at current level are shown. Faces intersecting with boundary are not in the figure (already propagated up). Second, the part of the dense region near to the boundary is not selected as a buffer because no intersecting face is within the $i - neighbours$ distance.

the field boundary are selected. Then the 'read-only' buffer based on the i-neighbours of these faces is computed. Since the boundary intersecting faces and buffer faces are propagated up and generalization is only performed in the non-buffer regions less work is done. As a result all generalization happens in smaller region – in the area that is shrunk inwards. When the field is processed, all objects from the 'read-only' buffer are propagated up.

§ 5.5 Statistics

We have tested our method with multiple datasets that differ in content, size and type of features. The details of these datasets (Figures 5.4 and 5.5 give an illustration) are as follows:

- CORINE Land Cover dataset for an area of the United Kingdom and Ireland with approximately 100,000 faces and an area around Estonia with approximately 130,000 faces both intended to be used at a 1:100,000 map scale (Figure 5.5a).
- A road network dataset of the Netherlands (in Dutch: Nationaal Wegen Bestand). This dataset is intended for use at a 1:25,000 map scale. Note that the cycles/ areas of the road network (*i. e.* the 'space between the roads') have been used for creating a planar area partition and driving the generalization procedure (iteratively picking the smallest cycle of the road network to be removed, ultimately leading to removal of an road from the network, as described by Šuba et al. (2014b)). It contains approximately 200,000 of these areas.
- Dutch cadastral dataset (scale 1:1,000) for area of the province of Gelderland (The Netherlands) with approximately 880,000 faces. The geographic extent

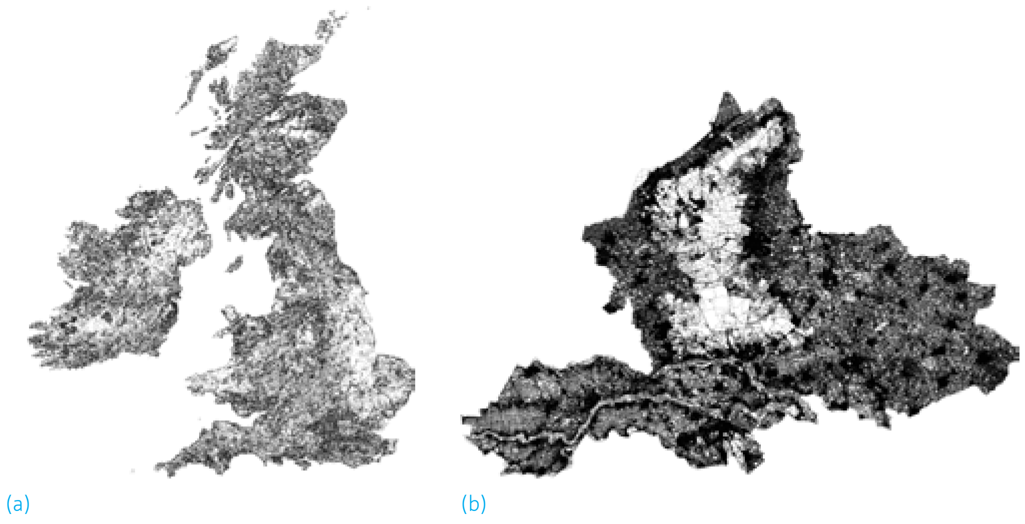


FIGURE 5.5 Two of the testing datasets: (a) CORINE and (b) cadastral map

of the data spans 130 km × 90 km. Although the generalization process is less meaningful for cadastral parcels, this dataset is a useful test dataset for our research as it is topologically structured and has been proven to be topologically clean and valid (Figure 5.5b).

All datasets have been processed in parallel (multiple fields at same time) at our research server with 16 CPU cores[1]. All vario-scale tables are stored in a PostgreSQL database management system extended with PostGIS[2].

The different datasets we processed show that the approach works, even for different types of input data (land cover data, road network and cadastral map). The successful processing with the road network dataset demonstrates that when it is important to have neighbouring features available (for generalization decisions), it is possible to shrink the domain inwards. With this approach we are still able to process fields in parallel, while each feature is just processed once.

To obtain insights into what is a reasonable field size for the smallest fields in the fieldtree, we ran the whole process (constructing fieldtree, distributing the objects, generalization field by field, finalization) with different sizes for the smallest fields in the fieldtree. We varied the sizes of these fields, based on the average size of a feature in the dataset. We used sizes varying from 20 to 160 times the average feature size. Figure 5.6 shows the resulting timings we obtained. The graph shows that as rule of thumb, we can set the size of smallest fields to 100 times the average feature size (*i. e.* around 10,000 objects in the smallest fields) and that the total runtime is then rather

1 HP DL380p Gen8 server (with two 8-core Intel Xeon processors, E5-2690 at 2.9 GHz, 128 GB main memory, and RHEL 6 operating system) with Disk storage (direct attached) with 400 GB SSD, 5 TB SAS 15K rpm in RAID 5 configuration (internal), and 2 × 41 TB SATA 7200 rpm in RAID 5 configuration.

2 PostgreSQL 9.3.4 and PostGIS 2.2.0dev

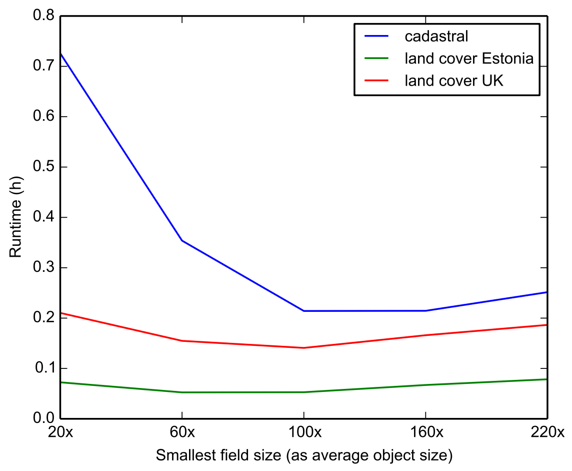


FIGURE 5.6 Processing time needed versus field size (smallest fields). Note that the smallest field size was determined as a multiple of the average feature size in the dataset.

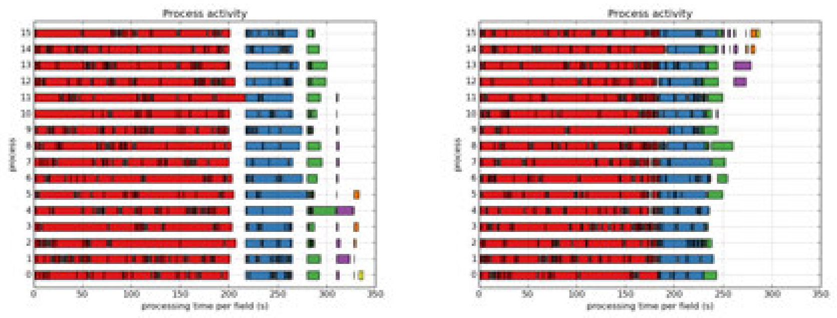
optimal. Furthermore, this graph teaches us that too small fields lead to a large overhead in processing: Many tables that contain a little bit of data cannot be generalized sufficiently for the next level (so no reduction will take place at a level, wasting processing time / copying data to the next level).

For the biggest dataset (the cadastral map with 880,000 faces) we investigated in-depth the runtime needed for the generalization step. In a sequential run this step took 7900 seconds (slightly more than 2 hours). We measured the time needed for parallel processing field by field. We also investigated whether this could be shortened by taking a different order of processing the fields. Figure 5.7 shows two possible scheduling strategies for distributing work in multiple parallel CPUs.

The first strategy, the 'level order' scheduling, takes care of all fields at the same level before the higher level is started. The second and more refined strategy, the 'parent child' scheduling, will schedule the parent field for scheduling if all its children are processed.

We assumed that the second scheduling option would significantly reduce the processing time, but this was a bit disappointing. A modest reduction was observed – from 340 to 290 seconds (15%). The main bottleneck remains near the top of the Fieldtree where most of the CPUs have to wait while only few processors are used. Note that the processing time of a field is dependent on the amount of data in that field. This might be used in a better scheduling strategy.

We found having processed the CORINE land cover dataset that the majority of the generalization work was performed towards the end of the whole process (40% of the runtime was devoted to processing the last field). After inspection, this turned out to be caused by some very large polygons remaining to the end of the process (these massive polygons with a lot of holes representing the land between all small settlements, which



(a) Level order scheduling (b) Parent child scheduling

FIGURE 5.7 Two different scheduling orders for the cadastral dataset. Every rectangle represents one field. Note that different colours correspond to the levels of the fieldtree (*i. e.* red corresponds to the fields with finest grid size). (a) Level order scheduling: The process waits until all fields of the same level are processed before processing of the next level starts. (b) The parent child scheduling: Fields are scheduled for processing when all its up to 9 child fields are done.

only fitted in the largest field of the Fieldtree). Apart from having a negative influence on the run time, these polygons will also have a negative impact on the cartographic result as these polygons will keep their full original amount of detail until when they are processed.

In order to assess the visual impact, we performed two following tests:

- We compared the result of the parallel version of the algorithm with the sequential run (original non-parallel approach); see Figure 5.8. The visual impression of both approaches are quite similar. The main difference is that the number of vertices is higher in the parallel version (counting showed that this is roughly twice the number of vertices). We assume this is due to number of lines fitting in the field, *i. e.* even longer lines fit in the field in the sequential run, hence more line processing could be performed.
- We selected three output maps at specific locations, in order to observe the effect of the regular grid on cartographic quality of the end result, see Figure 5.9. We chose three locations; *Slice A*, right after the lowest field level, *slice B*, in the middle of the second field level, *slice C* at the beginning of the third. The most visible 'seams' are right after previous level because almost no generalization took place on the field boundary compared to the generalized centre of the field. Note that user's viewport is smaller than any of the field of the grid. This means user would not have enough context information on the display to observe differences between generalized and less generalized regions.

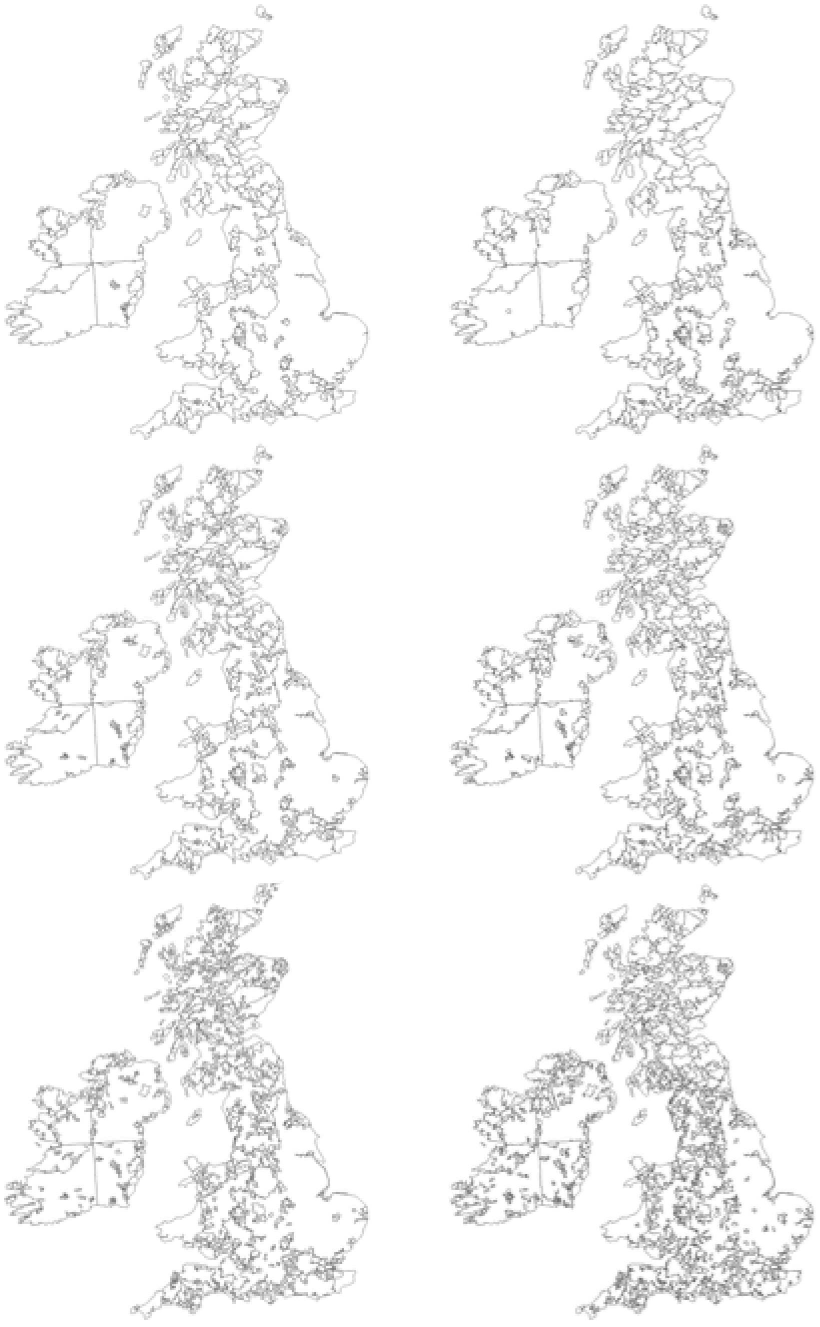
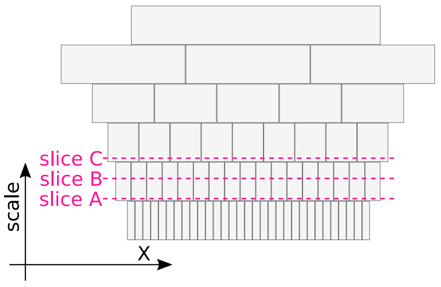
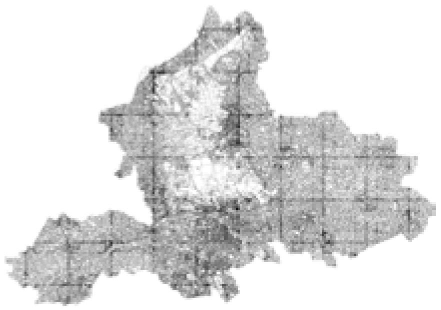


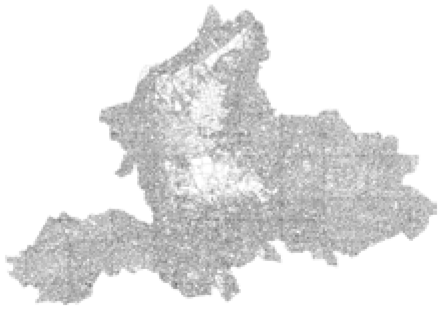
FIGURE 5.8 The visual comparison of a sequential run (in left column) and run in parallel (in right). The rows display different feature counts: 250, 500 and 1000 from top to bottom.



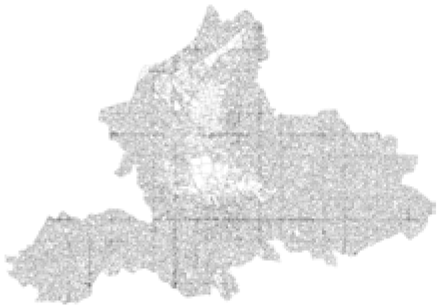
(a)Fieldtree – Side view for the cadastral dataset. Dashed lines indicates location of output maps captured in 5.9b, 5.9c and 5.9d.



(b)Slice A



(c)Slice B



(d)Slice C

FIGURE 5.9 The effect of regular subdivision of space on cartographic quality. An example of the cadastre dataset shows that 'seams' may be the most visible when inappropriate location in the Fieldtree is selected such as the beginning (b) or the end (d) of field level.

§ 5.6 Conclusions

An approach for automated map generalization to obtain a vario-scale data structure of massive datasets has been proposed and tested. We have described all steps of the process chain and we tested the approach with real data. It has been shown that our approach is generic and works for different types of input data such as land cover or road network data. We have tested with real world data , although we still plan to test the approach with a really massive dataset (*e.g.* topographic dataset of The Netherlands or CORINE for whole of Europe) which will not fit in main memory. We expect that larger dataset will benefit from having even more parallelism assuming the hardware is available, *e.g.* 32 processors. The design is scalable: each job gets its input data from the database and writes the output again to the database. So, the whole dataset does not have to fit in main memory. As long as the active jobs do fit in memory, the process should remain efficient. We have to explore this further. If we keep the same amount of main memory but increase the number of parallel processes, will these jobs still fit in main memory? If not, we should either make the average job size smaller or add more memory.

The Fieldtree is very appropriate as it is good start for divide-and-conquer, but also offers very useful multiple levels: What is not solved at the lowest level can be solved at higher (which glues partial solutions from lower level fields together). This fits very well with our approach to map generalization, but the divide-and-conquer approach using the Fieldtree may not be limited to map generalization only, and other spatial problems could benefit from this type of organization as well, *e.g.* obtaining an explicit topological data structure.

We have presented two approaches for scheduling the parallel work. We are aware of the fact that both of the scheduling approaches are still sub-optimal. An improved scheduling might be based on processing the largest fields (with respect to number of features contained) first and smaller later, and applying dynamic scheduling from a 'work-pool'.

Furthermore, we only carried out limited evaluation of the cartographic end results and as we found out with processing the CORINE land cover dataset with massive polygons, our approach is less robust against these polygons.

This shows that there are still some open questions to be answered. Our future work includes:

- Very large features survive until the end and are not generalized. These large objects and also their island objects are affected. In our tests typical cases are: large 'background' polygons, or very long infrastructure features, such as rivers and roads. Our proposal for a solution in this case: split the large features. This raises another interesting challenge: How to best split these large features into smaller (artificial) chunks.
- The effect of the regular grid on cartographic quality of the end result can be reduced. The 'seams' of the grid are caused by non-processed features because they lie on boundary of a field. Therefore we propose two options; a) to use two grids at every level (the second one shifted to the middle) and have two gener-

alization steps of 37,5% (instead of one 75% generalization), b) to have smaller steps between levels (and less generalization per level), result is more levels, but also more arbitrary off-sets.

- The proper size of the lowest fields need further investigation. What is a good balance between size of fields, running time and cartographic result? We showed that the running time depends on the field proportions and number and sizes of the features within. Too small fields lead to a lot of overhead in processing. Too big fields can lead to memory problems. This implies that a proper size of fields is crucial for obtaining a good cartographic end result. The size of fields is reflected in generalization quality. If the fields are big the 'seams' of the field may be visible, because the objects on the boundary could not be generalised for a long time while the rest of the field has been simplified a lot. By contrast too small fields will contain fewer objects which leads to fewer generalization options again leading to a worse result.
- The Fieldtree subdivides space regularly where the density of the original data is not considered. In some cases a large city is in one field (lot of generalization processing is needed) while the rest of the fields cover the rural regions where not much simplification is required. Then most of the resources are spent on the field containing the city while the other fields are already processed. A good strategy would be to locally deepen the fields at lowest level where high data density regions are reached. This would result in a slightly different Fieldtree: a structure of which the lowest deepened fields are not covering the whole domain any more
- An influence of the parallel creation compared to the traditional creation (based on global criterion for least important feature) on cartographic quality is not verified. For example is it possible to distinguish area processed by a different field? Does the user perception change when we use different/modified grid of the Fieldtree? Based on initial visual inspections we assume this is probably not the case, but should be verified with user tests.
- As the parallel creation is not based any more of the global criterion (of least important feature), the Fieldtree structure can be also used for the update of vario-scale structures: only the field with updated data needs to be reprocessed, and only if generalized result of field is significantly different, then also the parent(s) needs to be reprocessed (recursively).

6 Smooth zooming

Chapter 2 showed that current maps on the Internet are composed of the discrete set of LODs/scale pyramids with big changes of map content and representations. This can lead to confusion for the users when they navigate in the map. Therefore, a conceptual model (SSC) was proposed, see Chapter 3. We believe that, by capturing the whole generalization process in small smooth incremental changes, it is possible to achieve a better user experience *e. g.* when the user zooms in and out. To verify this hypothesis it is necessary: (1) generate a SSC dataset (2) develop a software prototype where a dataset can be used in its full potential and (3) perform usability test. Hence this chapter will describe the benefits of such as smooth representation in more detail. Section 6.1 gives an introduction to the problem and it suggests our solution. Section 6.2 covers the theoretical background, principle and example of smooth zooming. Section 6.3 presents possible conversion strategies to smooth representation. Section 6.4 explains the software prototype for possible usability testing. This is followed by our initial usability test, which was carry out addition to our plans. More specific, Section 6.5 defines all elements of the testing. Section 6.6 presents preliminary results, and Section 6.7 describes gained experiences. Then, Section 6.8 summarizes possible improvements for the future.

Own publications

This chapter is based on the following own publications:

- Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014a). An area merge operation for smooth zooming. In Huerta, J., Schade, S., and Granell, C., editors, *Connecting a Digital Europe Through Location and Place*, Springer Lecture Notes in Geoinformation and Cartography, pages 275–293. Springer International Publishing. ISBN: 978-3-319-03611-3.
- Šuba, R., Driel, M., Meijers, M., van Oosterom, P., and Eisemann, E. (2016a). Usability test plan for truly vario-scale maps. In *Proceedings of the 19th ICA Workshop on Generalisation and Multiple Representation, Helsinki, Finland*.
- Huang, L., Meijers, M., Šuba, R., and van Oosterom, P. (2016). Engineering web maps with gradual content zoom based on streaming vector data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:274 – 293.

§ 6.1 Benefit of smooth representation

Cartographic generalization is the process of transforming a map from a detailed scale to a less detailed scale. Only the end result of such a process is usually stored. However, for some applications the visual process of continuously changing the level of detail is important. Instead of discretely switching from one scale to another, a continuous transformation from source to target scale is preferable because it provides a better impression to the user.

An essential factor in map usage is how the user perceives a transition between map scales at different levels of detail in the course of zooming in or out. From the experiment carried out by Midtbø and Nordvik (2007) it is evident that sudden changes during zooming are distracting to the user, and may also result in losing track of the objects the user is interested in.

The classical solutions for zooming are based on a multi-scale approach, where every scale with different map content is stored separately. Then the zooming is effectively 'switching' from one map to another with sudden map content change. Attempts are made to relieve 'map shocks' by tricks such as graphic enlargement of the map layer before 'switching', blurring the map at the time of switching or map morphing, *i. e.* an animated translation from one map to another (Reilly and Inkpen, 2004, 2007).

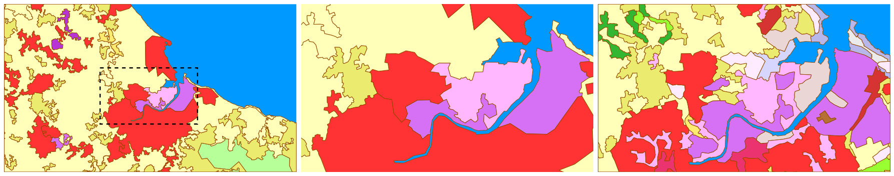
Despite some useful efforts presented so far, see Section 2.3, there is no optimal solution yet. This brings new challenges but also new demands which should be reflected in map generalization and its requirements. Based on the current solutions, we can identify two following directions to introduce more gradual change of the scale:

- Generating smooth content – The purpose of the map is considered in the data generation. The map generalization is perceived as a production of a sequence of successively more generalized maps. The sequence can then be shown to the user. The characteristic of data should be also consider for data transfer; *e. g.* (Sester and Brenner, 2005; Chimani et al., 2014).
- Using graphic techniques – The original data and data in transfer are not modified. Additional graphic techniques are used on top of the map solution. Most of the time well-known techniques from the field of computer graphics are used such as morphing, blending between the layers, transparency and initialising, *e. g.* (Reilly and Inkpen, 2007; Nöllenburg et al., 2008; Danciger et al., 2009).

Our vario-scale solution with representation of the generated sequence of the generalization steps fits into the first category, see description of the concept in Chapter 3, which provides solid base for smooth user experience. However, for optimal smooth user experience both smooth content and graphical techniques are important. Additionally, the zooming operation can be performed and implemented in a variety of different ways. Therefore, the following explanation describes the zooming functionality used in this chapter.

§ 6.2 Smooth zoom aspects

Let m_0 and m_1 be the two maps that will be shown while performing a zoom operation. The initial map m_0 shows coarse data at a specific scale (small scale). Map m_1 will contain more detailed data (larger scale), since it covers a smaller geographical area. Note that both the area and the content of the maps m_0 and m_1 are different.



(a)Initial view m_0. Dashed rectangle shows the final extend of the map m_1.

(b)Original map m_0 where only graphical zoom has been applied.

(c)Map m_1 where the content zoom has been fully applied.

FIGURE 6.1 User experience of a zoom in operation. Figure (a) is original map. The rectangle shows the area zoomed in upon by the user. Figure (b) where the content of the initial coarse map is first graphically enlarged and then replaced by the more detailed map (in one go, thus not in small incremental steps). Figure (c) Map after the *rescaling* and the *content zoom* operations have been applied.

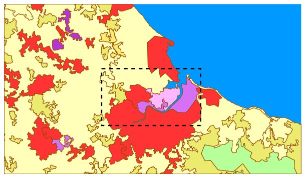
The transition from m_0 to m_1 is what we call a 'zoom in' operation. During this transition two operations are performed: For the first aspect of the transition, called *rescaling*, the objects shown on map m_0 are graphically enlarged (scaled, translated). Only the region corresponding to m_1 is shown (clipped) after this enlargement (no new content is yet retrieved or shown). The second aspect, called *content zoom*, changes the objects of the map (the content). The already graphically enlarged region of map m_0 changes its contents to become map m_1, see Figure 6.1. A zoom out operation applies the same steps as the zoom in operation, but in reverse order: instead of enlarging the map it shrinks it. Starting with map m_1, it first shrinks the objects graphically to the extent of m_0. Then the content is changed to become map m_0. Additional *graphic techniques* (*e. g.* blurring, morphing or making objects more transparent) can be applied for both (*rescaling and content zoom*) aspects in the transition. Note that a zoom operation consists of one *rescaling*, one step of *content zoom* and none or more *graphic techniques* all together integrated in one final map (*e. g.* m_1).

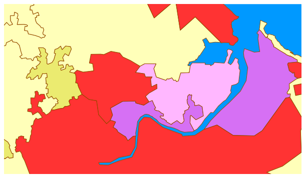
Figure 6.2 illustrates an alternative that, with a smaller difference in the content between map m_0 and m_1 the *content zoom* step is perceived to be changed more gradually (less of a shock). Therefore, we can apply the *content zoom* operation multiple times, changing the map content only bit by bit (leading to more temporary maps, showing the transitioning of the map content in small steps from m_0 to m_1 , *e. g.* m_0, m_0', m_0'', m_0''',... , m_1), and thus progressively refining the map. This leads to a very smooth transition from one scale to another -– hence the term *smooth zoom*.

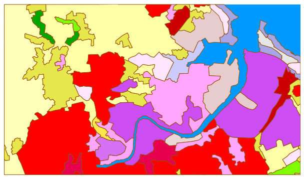
To conclude, we can define *smooth zoom* operation as following: After one user request (*e. g.* zoom-in) a series of frames is generated with small differences per frame w.r.t. both *graphic* and *content zoom* aspects. Then, one *smooth zoom* operation consists of one *rescaling*, *content zoom* of n steps (where $n > 1$, *e. g.* m_0', m_0'', m_0''', m_1) and none or more *graphic techniques*. This is integrated in n final maps in order to achieve a very smooth transition from one scale to another.

§ 6.3 Generating 3D Space-Scale Cube

In the previous section we gave detailed description of the method which could provided better user map impression. It is based on combination of *graphic techniques*,


(a)Initial view m_0. Dashed rectangle shows the final extend of the map m_1.


(b)Original map m_0 where only graphical zoom has been applied.


(c)Map m_1 where one incremental step of the *content zoom* has been applied.

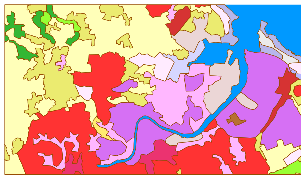

(d)Map m_1 where the *content zoom* has been fully applied.

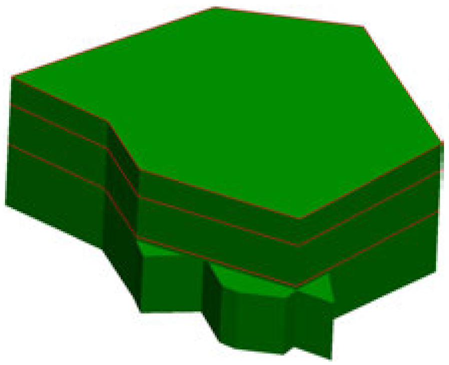
FIGURE 6.2 User experience of a smooth zoom in operation. The content of the initial coarse map is first graphically enlarged (shown in Figure (a) and (b)) and then replaced in incremental steps, leading to the refined map (shown in Figure (c) and (d)).

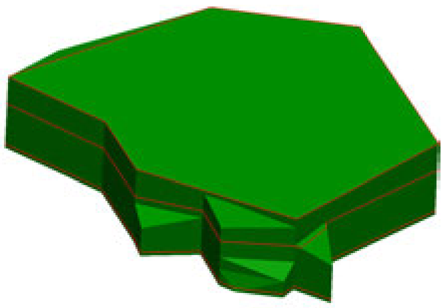
rescaling and transition over the sequence of maps with small changes in content. However, an issue still remains; how can such a map sequence for *content zoom* be generated?

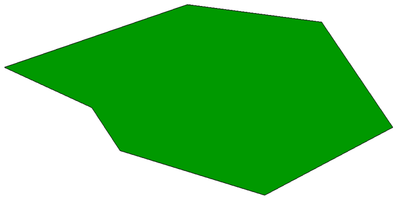
It has been investigated in (van Oosterom and Meijers, 2011b), (see Section 3.4 on page 28) where the tGAP structure delivers vario-scale data and can be used for gradual content zoom in the form of SSC representation. The structure has very significant advantages over existing multi-scale/multi-representation solutions (in addition to being truly smooth vario-scale): (a) due to tight integration of space and scale, there is guaranteed consistency between scales (it is one integrated space-scale partition), (b) since map features are represented as volumetric data, which are 'sliced' to produce maps, it is relatively easy to implement because this is well-known method from 3D computer graphics, (c) it provides a compact (good for storage, transfer and CPU use) and object-oriented encoding (one higher dimensional representation for a single object over the complete scale range).

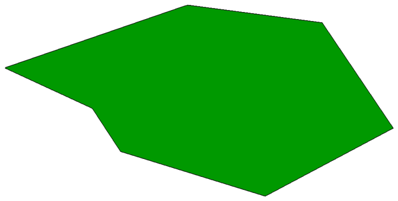
We focus on transition of vector data in a representation convenient for content zoom captured in tGAP structure to be stored in a SSC, investigating generalization operations such as merge, split, line simplification to do so. Figure 6.3 illustrates attempt at implementing a line simplification algorithm that generates smooth output for one object. In this section will focus on the merge operation (aggregation) as the most dominant operation in tGAP structure creation and we will present three algorithms for such a transition. It is important remind that tGAP/SSC exists in two versions: classic and smooth, see Section 3.4.

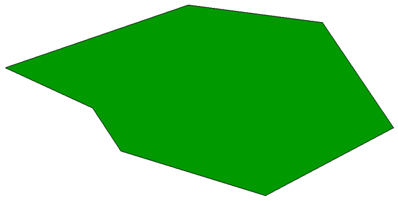
We aim at smooth representation in the smooth version of tGAP/SSC, where all changes result in a smoothly changing 2D map: a small change in the map scale means a small change in the geometry of the resulting map, see Figure 3.8b on page 30. Figure 6.4

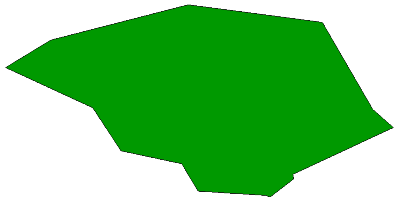

(a)3D perspective view (*with* sudden boundary simplification). The lower part of the object is result of previous step. It is used as input for simplification.

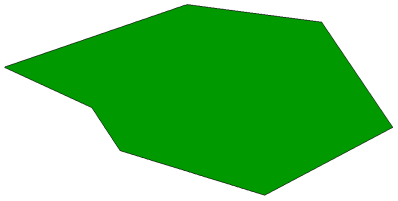

(b)3D perspective view (*with* smooth boundary simplification)


(c)(less detailed) – without smooth simplification


(d)(less detailed) – with smooth simplification


(e)without smooth simplification


(f)with smooth simplification


(g)2D horizontal slice (detailed) – without smooth simplification

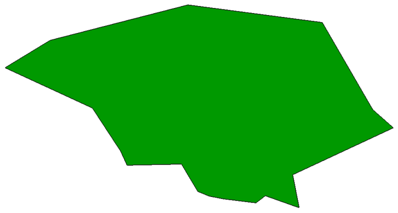

(h)2D horizontal slice (detailed) – with smooth simplification

FIGURE 6.3 Two Space-Scale volumes with (right column) and without (left column) smooth line simplification and some derived slices. The red lines indicate the position of the slices. The boundaries of the object are simplified with the Visvalingham Whyatt algorithm (Visvalingam and Whyatt, 1993), leading to a coarser object. The order in which the vertices are removed from the line by the algorithm are stored in a binary tree structure (the so-called Binary Line Generalization tree, BLG tree). This tree structure provides enough information to construct a 3D volume for the object.

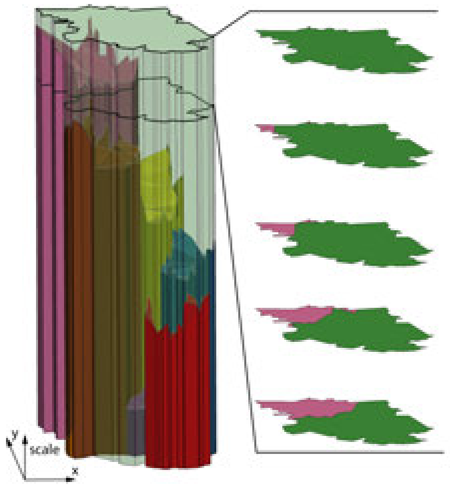


FIGURE 6.4 A small subset of CORINE Land Cover dataset in smooth tGAP with arbitrary slices. The colours are randomly assigned. Shows 7 area objects (at the most detailed level). The horizontal slices illustrate the last step of the tGAP structure where the pink object is merging with green one.

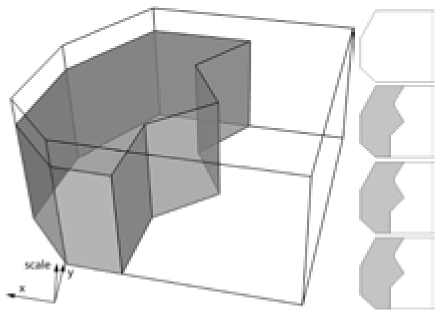
provides an example of smooth representation for small subset of CORINE Land Cover dataset. The arbitrary slices in Figure 6.4 demonstrate the final impression from a vertical shift of the slice plane.

The remainder of this section is organised as follows: First, the basic principle of smooth-merge operation is described in §6.3.1. Second, the requirement for such a conversion are covered in §6.3.2. Third, designed algorithms for the operation are demonstrated in §6.3.3 followed by §6.3.4 describing the storage efficiency. Finally, §6.3.5 presents a final comparison of the algorithms.

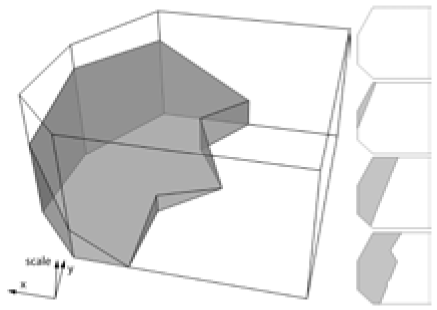
§ 6.3.1 The principle of smooth-merge operation

The creation of the tGAP structure is based on the merge operation of the least important object which is called the $loser$, with its most compatible neighbour, called the $winner$. Figure 6.5a presents such an operation in a classic tGAP structure, where the white winner merges with the grey loser and creates the growing white object. Figure 6.5b presents the same process in the smooth tGAP, which will be termed the smooth-merge operation. Figure 6.5b shows that any arbitrary horizontal slice leads to a new 2D map. If the slice plane is moved up, then the white winner grows and the grey loser shrinks. All the geometry changes gradually.

During the transition to the smooth tGAP structure, some objects could be deformed or misrepresented and the resulting map (slice of the smooth tGAP representation) can be confusing. For instance, a single object representation could break into multiple parts, see the white object in Figure 6.6. Such spurious multiple parts cause a transitory increase in the number of objects in the map and result in a degradation of the



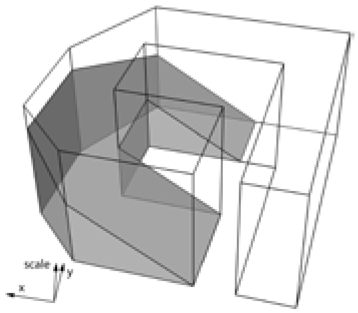
(a)The merge operation in classic tGAP



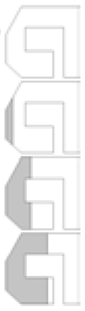
(b)The merge operation with arbitrary slides in smooth tGAP

FIGURE 6.5 The merge operation in classic tGAP in (a) and smooth tGAP in (b). The white winner takes space from the grey loser. Slices are shown at four levels, from top to bottom in the image: from high, to low.

quality of the map. Therefore, our intention is to ensure that area features do not break into discontinuous parts.



(a)Space scale view. Only shared boundary between objects is shaded.



(b)Slices

FIGURE 6.6 More complicated shapes with arbitrary slices in smooth tGAP. The middle and low slices in (b) show two parts of the same object (white winner).

§ 6.3.2 Requirements

An important challenge of our research was to transfer objects from classic tGAP to the smooth tGAP represented in SSC where horizontal faces, causing abrupt changes, do not exist. A smooth-merge operation was developed. This should compute in 3D a set of tilted faces representing the boundary surface between the loser and the winner. We identified the following requirements for good smooth merge operations (each one supported by its own motivation):

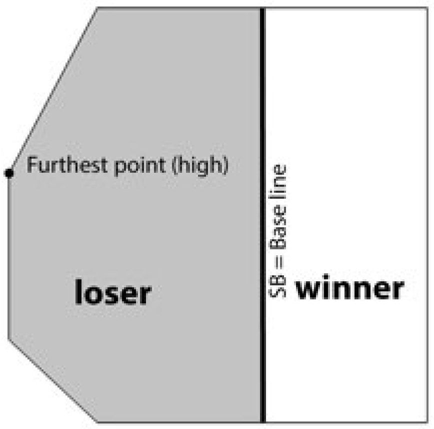
I. Topologically correct in 3D – The input 2D map is topologically correct and forms a 2D partition of space, the resulting 3D representation should also be topologically correct and not have gaps, overlaps or intersections
II. No new points, that is no new x, y coordinates – The construction of tilted faces make use of the already existing geometry. It follows from one of the main aspects of the vario-scale approach, which is to minimize the redundancy of data. Only the new edges and faces are created using existing geometry.
III. No horizontal faces – This follows from the definition of the smooth tGAP structure mentioned above as horizontal faces cause sudden changes.
IV. No vertical faces between winner and loser – The winner should gradually take over the area of loser, but vertical face means nothing happens here for a while.
V. No multi-parts – Spurious multi-part objects (which should be single part), add confusion to the map.
VI. Constant steepness of tilted face/faces – The shared tilted boundary composed from multiple faces with different steepness would result in the effect that some parts of the loser merge much faster than others during smooth zoom. The steepness of the faces should be as constant as possible.
VII. Optimal shape of shared boundary – The shared 3D surface boundary can consist of multiple faces. There often exist more configurations of faces, and each defines a 3D surface of the shared boundary. The configuration that gives the best merge impression to user should be selected. That is, a natural looking boundary should remain during the smooth zoom (slicing operator).

The list above contains three 'hard' and four 'soft' requirements. The first three requirements will always be guaranteed by the algorithms. It is not possible to guarantee the all other requirements, as they are sometime contradictory and they are therefore classified as soft requirements. It will be tried to optimized these in a well balanced manner. Note that the different 'soft' constraints may be competing; *e. g.* constraint 6 has preference for a single plane (equal steepness), but may result in multi-parts; see Figure 6.6 . The hard requirements guarantee functionality and they are crucial to finding the solution. The soft requirements are aesthetic requirements which are more like recommendations for a good map recognition (readability). We would like to fulfil them. However, they are not to be strictly satisfied, *e. g.* In some cases a loser can be composed of multi-parts.

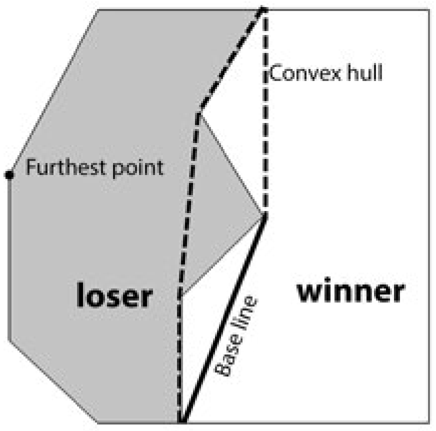
§ 6.3.3 Three methods

Three different algorithms for smooth-merge operation have been created and implemented to satisfy the hard requirements and optimize the soft requirements: the 'Single flat plane', the 'Zipper' and the 'Eater'. The difficulty of implementation with these algorithms ranges from trivial, with the 'Single flat plane'; to more complex, with the 'Zipper'; to rather complex, with the 'Eater'. Based on the various types of map objects – the thematic classification and shape, which may result in different emphasis on the soft requirements, it should be possible to select the most suitable algorithm.

SINGLE FLAT PLANE The first implemented algorithm was 'single flat plane'. It originated from idea that smooth-merge can be represented by simple single flat plane of



(a) A case where the base line is same as the shared boundary.



(b) A case where the shared boundary is not a straight line. The base line is the longest edge of the convex hull (dashed line) of the shared boundary located in the winner.

FIGURE 6.7 Definition of the Base line for the 'Single flat plane'.

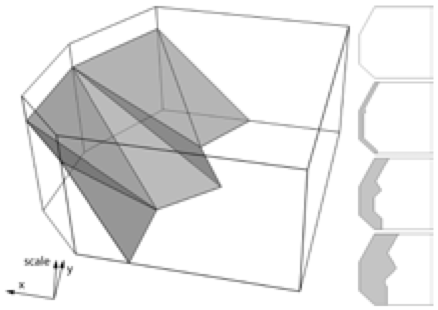
constant steepness as illustrated in Figure 6.5b. In principle, the loser (the face that has to be removed) can always be removed with a single flat plane. The plane is defined in 2D by the shared boundary, the edge(s) between the winner and the loser, and the furthest point of the loser from the shared boundary; see Figure 6.7a. Then in the 3D the plane is lowest at the shared boundary and highest at the furthest point. As it is a single flat plane, linear interpolation can be applied for calculating the height at any other point on the share boundary surface between the winner and the loser. Figure 6.5b shows the final 3D representation of smooth merge using the 'Single flat plane' algorithm. In context of the smooth-merge operation the shared boundary is used, where the merging starts and the distance from the shared boundary to every point of the loser is calculated to find the point most far away. If the shared boundary is not a straight line; see Figure 6.7b, then the concept of a base line is introduced. The base line is an 'approximation' of the shared boundary and it is used for measuring distances and finding the furthest point. Three ways of finding the base line are investigated:

- The base line is the longest edge of convex hull of the shared boundary located in (or on the boundary of) the winner; see Figure 6.7b.
- The base line is the edge of the smallest rectangle around the loser which has biggest overlap with the face of winner (and has loser completely at other side).
- The base line is result of a best-fit line technique known as Eigenvector line fitting to obtain the orientation of the base line (van Oosterom, 1990, p. 61). There are two options: use all points of the loser or just the points of the shared boundary. Then this Eigenvector line is translated to make sure the loser is completely on one side (and touching the line) with the winner on both sides, but preferably with the largest part opposite the loser.

The first approach, the longest edge of convex hull of the shared boundary has been used for our implementation mostly because it gives the best results in initial testing.

The most serious limitation of the 'Single flat plane' approach is orientation. There always exists only one direction where the plane can be oriented. However, in some cases the winner and the loser can have more shared boundaries, *e. g.* the boundary between the winner and the loser is broken by another polygon. In such a case, the decision where the tilted plane should be placed must be made. The situation is even worse when the loser lies inside the winner, where no good quality solution exists. Also in case of a very long and faceted boundary, the base line will not be able to represent this well. Some vertical faces are needed to make the single flat plane fit into the 3D SSC. Note this is also the case when base line fits rather well, only then the vertical faces will not need to be very high. The vertical faces result in the effect that at their locations for a while, nothing changes while other parts the winner is already taking space from the loser. In addition to this, users may find objects modified by 'Single flat plane' disturbing while they use smooth zoom. This is because the boundary between the loser and winner may become a straight line during transition which can be perceive as artificial sweep line going from one side to another.

Despite the above mentioned limitations of the 'Single flat plane' algorithm, it can be very effective for simple convex polygons where a simple shared boundary exists. The computation of the base line, together with a definition of the tilted face, is trivial. From the smooth zooming point of view, the 'Single flat plane' has the advantage of the winner face growing with constant speed. These aspects make the 'Single flat plane' algorithm a good candidate for processing simple convex faces, such as rectangular buildings.



(a)Space scale view. (b)Slices

FIGURE 6.8 The 'zipper' algorithm. The white winner taking over space from the gray loser. (b) shows slices at four levels from top to bottom.

ZIPPER The second approach is based on the decomposition of the loser polygon into triangles. These triangles will then in 3D become the tilted surface of the shared boundary. The segments of the boundary of the loser are classified into two types: (1) The Shared Boundary (SB), which is the boundary between the winner and the loser and (2) The Opposite Boundary (OB), which is the remaining part of the boundary of the loser, holes included. Figure 6.8 shows the final 3D representation. It is assumed that the tilted faces should be tilted from SB (low) to OB (high).

Before we designed the 'Zipper' algorithm, we first investigated another method for finding a good solution satisfying as many requirements as possible. We call it 'Delaunay triangulation and flipping'. It starts with a Delaunay triangulation of the loser, giving the fattest triangles possible (satisfying the soft requirements). Then it traverses the triangulation and flips edges which might provide a better configuration of the triangles more satisfying the requirements. This method is computationally expensive and does not guarantee the best solution, because a local flip of the triangles may not lead to the global optimal solution.

Therefore, another method, the 'Zipper', was implemented. The hard requirements remain valid during the whole process: topology is correct, no new vertices (x, y coordinates) are created and no horizontal faces are generated. Note that horizontal faces appear when a triangle connects vertices on SB only or on OB only. The other optimization rules have been met:

- Every triangle connects SB and OB, which means that at least one vertex lies on the SB and at least one other vertex lies on the OB;
- The triangles should have a low aspect ratio - the aspect ratio is the longest side divided by the shortest altitude of triangle (Shewchuk, 1996). It guarantees that the triangles have no sharp angles;

These additional rules guarantee that we get a good solution according to the hard and soft requirements. Another optimization rule, not further explored could be: the two edges that run from SB to OB should be as much as possible of equal length, making sure that the speed of transition is as nearly as possible equal.

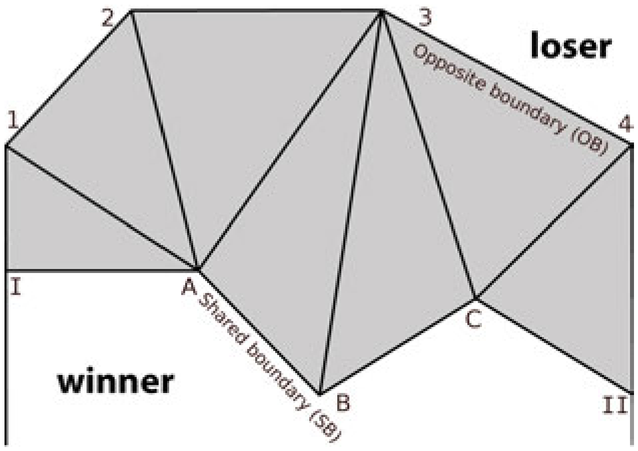


FIGURE 6.9 Principle of the 'Zipper'. The optimal solution, only the final connections are present.

Figure 6.9 describes the principle of the 'Zipper' algorithm. One can imagine the process as a walk along the SB and OB simultaneously, creating of a connection from one vertex at SB to another vertex at OB, if possible. When the connection is created, a new triangle is defined. The process starts at the junction of SB and OB (vertex I in Figure 6.9) and ends in another junction (vertex II in Figure 6.9). For every edge, connecting SB and OB, the aspect ratio is computed. Then the process can start from the other side of the SB. The optimal solution is defined as one where the whole area of the loser is processed and where the sum of aspect ratios is minimal.

Figure 6.10 presents an alternative visualization of the process as for a step there can be two options; 1) forward SB, 2) forward OB. The process can be represented as a graph where every connecting edge is a node in the graph. The vertical axis corresponds to the index of SB. The horizontal axis corresponds to the index of OB. Only moving to the right or moving downwards in the graph is possible as the algorithm to create triangles either takes a step on SO or OB, but not both as this would result in a quadrangle. The weight of a graph edge is defined as the aspect ratio of the triangle associated with the graph edge connects.

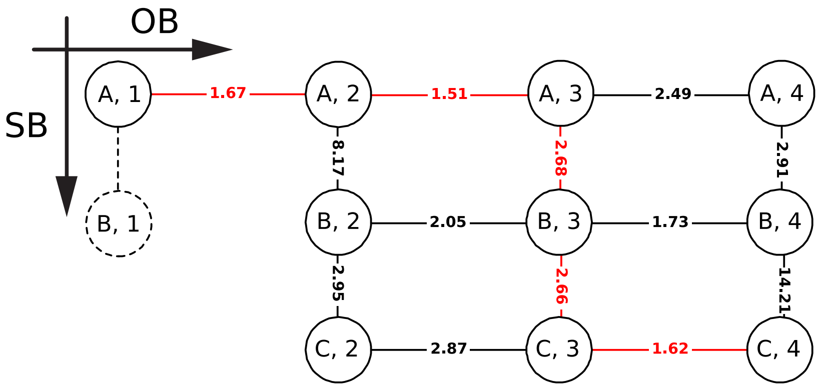
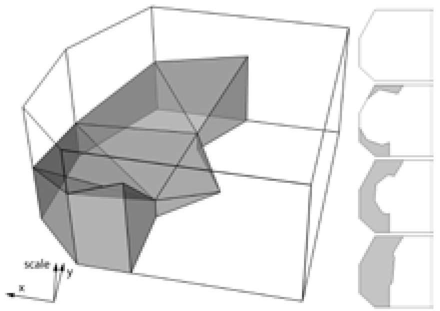


FIGURE 6.10 Graph representation of the 'Zipper' approach for configuration in Figure 6.9. The 'zipping' starts at triangle (I, A, 1) towards (II, 4, C). The process can also run in the opposite direction. The best solution is presented in red. Note that graph note (B, 1) (dashed) is not possible as triangle (B, A, 1) would be outside the loser.

The process starts in the upper left corner and finishes (if a solution exists) in the bottom right corner. If a solution exists, then a connection between the upper left and the bottom right corner exists.

The graph representation can be used for optimization. The best solution is the solution with the lowest total sum of aspect ratios of triangles (corresponding to graph edges). Basically, it is the cheapest path through the graph. As it can be observed, the edges with the lowest ratio tend to lie mostly on the diagonal of the graph, see Figure 6.10. This means that for an optimal solution it is sufficient compute the diagonal connection from the upper left to the bottom right corner and then if such a connection does not exist one can start to compute other nodes further from diagonal. If there is no connection between the upper left and the bottom right corner this means that it is not possible to process the loser in a single step, one of the faces of the loser must be split into more pieces. Note that in such a situation the other algorithm ('Delaunay triangulation and flipping') also cannot find a solution as there is no solution without splitting loser in multiple parts. Only certain types of polygons (all nodes on OB are visible from nodes of SB) can be processed directly. This always the case with convex polygons, but also for certain type of concave polygons; *e. g.* relatively long polygons with visibility between the two parts of the boundary SB and OB (*e. g.* road or river polygons without side branches, which should be treated as separate objects).

Where it is not possible to process the polygon in this way, the graph can suggest where a split can take place. The associated vertex of the graph nodes, where the connection fails in most cases, can be a candidate for splitting. Unfortunately, with splitting a number of possible solutions arise and for the overall optimal solution many or all of them should be checked. Therefore, another algorithm, which is called the 'Eater' was developed. It will be presented in the next section offering a general solution for arbitrary polygons without the need to split either feature.



(a)Space scale view. (b)Slices

FIGURE 6.11 Tilted faces of loser processed by the 'Eater'. (b) slices at four levels, from top to bottom in the image: from high to low.

EATER The third approach, which is called the 'Eater' provides a solution to any arbitrary polygon (with holes, concave, or multiple shared boundary parts). It is based on a triangular tessellation of the polygon and ordering the triangles into a gradual change from low to high. Figure 6.11 shows the result. The Delaunay triangulation is the initial step, where the face of the loser is tessellated. Then finding the starting triangles for the walk takes place. The triangles which have two edges of shared boundary are selected as starting triangles if any exist, and added to the so-called *active set*. These triangles are processed first which makes the shared boundary less 'curvy'. Note that when processing such a triangle in 3D there are two options how to set relative heights: (1) Keep both edges of SB completely at the lowest start height, or (2) keep only the shared node of these two edges at the lowest start and other two nodes at the first height step. The first option will result in a horizontal triangle and therefore violates a hard requirement. Therefore, the second option is selected, which has only the drawback of introducing some vertical triangles, but this is 'just' violating a soft constraint and not a hard one. If there are no triangles with two edges in SB, then all the triangles which have one edge in SB are selected as starting triangles and these add into the *active set* (and both nodes of involved edge get fixed height at start level).

The Algorithm Eater depicts the process steps to defined the triangle in 3D space, fitting into the SSC topology. Table 6.1 presents the run of the algorithm for the case captured in Figure 6.12 results in Figure 6.11.

In general, this algorithm guarantees a solution for every object, the face with multiple shared boundaries included. With this algorithm it is always possible to convert a dataset in classic tGAP representation into a smooth tGAP representation, where no horizontal faces exist. The price for that is some vertical faces and a chance of multiparts. The comparison and conclusion will follow in Section 6.3.5.

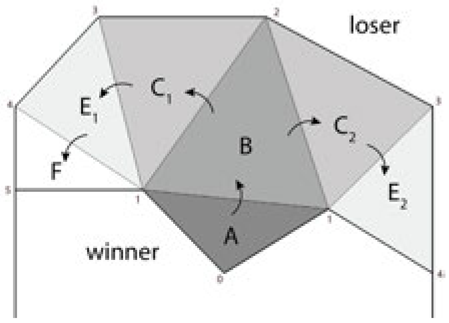


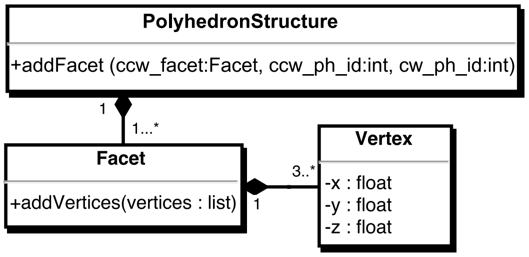
FIGURE 6.12 Principle of the ‘Eater’. The triangles are walked from dark to light grey. The numbers present relative z value.

Algorithm Eater

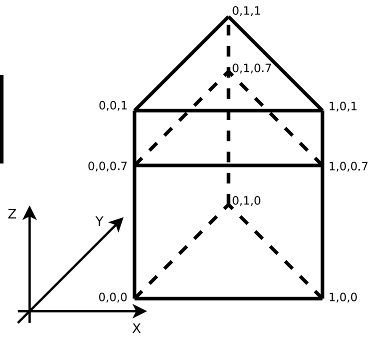
```
Require: All triangles of tessellated polygon
 1: All triangles set to $not - visited$
 2: Define starting triangle and put it to *active set*
 3: Set *relative height* to $1$
 4: *Next active set* $= \emptyset$
 5: while *active set* not empty do
 6:     Pop the first triangle from the *active set*.
 7:     for all vertices of triangle do
 8:         if vertex does not have height set then
 9:             Set vertex height to *relative height*
10:     Add all $not - visited$ neighbours of triangle into the *next active set*.
11:     Set the triangle to $visited$
12:     if *active set* is empty then
13:         *Relative height* $+1$
14:         *Active set* $=$ *next active set* and *next active set* $= \emptyset$
15: return A set of triangles with heights for all vertices
```

Step in algorithm	1. Iteration		2. Iteration		3. Iteration		4. Iteration		5. Iteration		6. Iteration		7. Iteration	
6	A		B		C_1		C_2		E_1		E_2		F	
10	$[]$	$\{B\}$	$[]$	$\{C_1, C_2\}$	$[C_2]$	$\{E_1\}$	$[]$	$\{E_1, E_2\}$	$[E_2]$	$\{F\}$	$[]$	$\{F\}$	$[]$	$\{\}$
14	$[B]$	$\{\}$	$[C_1, C_2]$	$\{\}$	$[C_2]$	$\{E_1\}$	$[E_1, E_2]$	$\{\}$	$[E_2]$	$\{F\}$	$[F]$	$\{\}$	$[]$	$\{\}$

TABLE 6.1 demonstrates the principle of Algorithm Eater and shows the sequence of the traversal of the triangles. The ‘Eater’ ends with the triangles ordered, and a relative height assigned to every vertex. The elements in square brackets refer to the *active set*, the elements in curly brackets refer to the *next active set*.


PolyhedronStructure
+addFacet (ccw_facet:Facet, ccw_ph_id:int, cw_ph_id:int)
1
1...*
Facet
+addVertices(vertices : list)
3..*
1
Vertex
-x : float
-y : float
-z : float

(a)UML diagram for the structure.



(b)An example of polyhedrons.

```
ssc = PolyhedronStructure()
#bottom
ssc.add_facet(((0,0,0),(0,1,0),(1,0,0)),-1,1)
#side 1.1
ssc.add_facet(((0,0,0),(0,0,0.7),(0,1,0.7),(0,1,0)),-1,1)
#side 1.2
ssc.add_facet(((0,0,0),(1,0,0),(1,0,0.7),(0,0,0.7)),-1,1)
#side 1.3
ssc.add_facet(((1,0,0),(0,1,0),(0,1,0.7),(1,0,0.7)),-1,1)
#middle
ssc.add_facet(((0,0,0.7),(0,1,0.7),(1,0,0.7)),1,2)
#side 2.1
ssc.add_facet(((0,0,0.7),(0,0,1),(0,1,1),(0,1,0.7)),-1,2)
#side 2.2
ssc.add_facet(((0,0,0.7),(1,0,0.7),(1,0,1),(0,0,1)),-1,2)
#side 2.3
ssc.add_facet(((1,0,0.7),(0,1,0.7),(0,1,1),(1,0,1)),-1,2)
#top
ssc.add_facet(((0,0,1),(1,0,1),(0,1,1)),-1,2)
```

(c)The snippet of Python code creates two simple objects from (b) in the structure.

```
v 0 0 0
v 0 1 0
v 1 0 0
v 0 0 0.7
v 0 1 0.7
v 1 0 0.7
v 0 0 1
v 0 1 1
v 1 0 1
g 1
f 1 2 3
f 1 4 5 2
f 1 3 6 4
f 3 2 5 6
f 6 5 4
g 2
f 4 5 6
f 4 7 8 5
f 4 6 9 7
f 6 5 8 9
f 7 9 8
```

(d)An example from (b) outputted as *.obj file.

FIGURE 6.13 An UML diagram and corresponding example of the usage for 3D topological structure.

§ 6.3.4 3D storage of small dataset

The whole dataset in smooth tGAP has been stored explicitly as polyhedral volumes in a 3D topological structure, see Figure6.13. The structure records topology for a set of 2-manifold faces in 3D. The core of the structure is a list of vertices, where scale attributes are used as z values in the 3D representation. Then all polyhedrons with their facets refer to the vertex indices. Note that shared vertices among the objects are always represented just once, not duplicated.

The number of resulting elements can give us some idea how efficient the storage is. The small subset of CORINE Land Cover dataset (7 faces) is used as an input; see Figure 6.4. It contains 676 vertices and 15 records of faces in classic tGAP structure. Because of its general applicability the 'Eater' algorithm was used for converting data stored in classic tGAP into the smooth representation.

This small example is represented by 989 vertices (which is more than the original 676 vertices as some of these have multiple counterparts at different height levels; however, there are no new x, y coordinates introduced) and 684 faces, consisting of: 289 triangles – the tilted faces of shared 3D boundaries between winners and losers, and 395 vertical rectangular faces (normal boundaries, which are not the result of a smooth merge). Finally there are 7 horizontal bottom faces and 1 horizontal top face.

When all objects of the dataset are converted to polyhedral volumes stored in SSC, a map can be obtained by horizontal slice. The continuously changing map can one imagine as gradually moving the slice plain from the top of the cube downwards. All changes result in a smooth changing 2D map: a small change in scale leads to a small change in map content.

One of the challenges here is the explicit boundary calculation of such a slice. This is computational intensive and can be very expensive due to the polyhedras' complexity.

As an alternative to this, we converted the smooth tGAP dataset to tetrahedrons (a tetrahedron is a geometric object composed of four vertices and four triangular faces) for visualization during development (Šuba et al., 2013). 1734 tetrahedrons have been stored for example above, see Figure 6.4. The ideal behind this was simple; the slicing can be done more efficiently because to make a slice of set of the tetrahedrons is much easier than making a slice of arbitrary 3D polyhedral objects.

However, the conversion to tetrahedrons adds extra computational step into the process workflow and it did not get significant speed improvement for the slicing. Therefore, we used different approach where full potential of graphic hardware can be used. Section 6.4 will cover more of this in detail.

§ 6.3.5 Comparison of three algorithms

Table 6.2 summarizes the possibility of smooth-merge operation for three algorithms, the 'Single flat plane', the 'Zipper' and the 'Eater'. Some criteria on the left side of the table come from soft requirements. Each of these criteria can be quantified and based on this the different algorithms are compared, however not all criteria are of equal importance when choosing the best solution for specific situation.

First, the 'Single flat plane' offers easy computation, constant speed of merging and it always creates just one tilted plane. The main disadvantages are defining the orientation of the tilted plane where the shared boundary is rather complex or more shared boundaries exist. The algorithm also does not try to avoid multi-parts and the final impression of merging may look artificial – a single straight line 'sweeping on the screen'. A low score has been given to 'Always has solution', because it in some cases the solution is not meaningful, *e. g.* where the winner lies inside the loser. Despite those limits the 'Single flat plane' algorithm can be very effective for simple convex rectangular polygons where only one shared boundary exists, such as the building in Figure 6.14.

Second, the 'Zipper' offers a more generic solution. It works for all convex and even for some concave polygons, this depending on visibility between SB and OB. It fulfils soft requirements and finds the optimal solution, if it exists; which results in good impression of merging and minimal risk of multi-parts. Higher computational complexity can

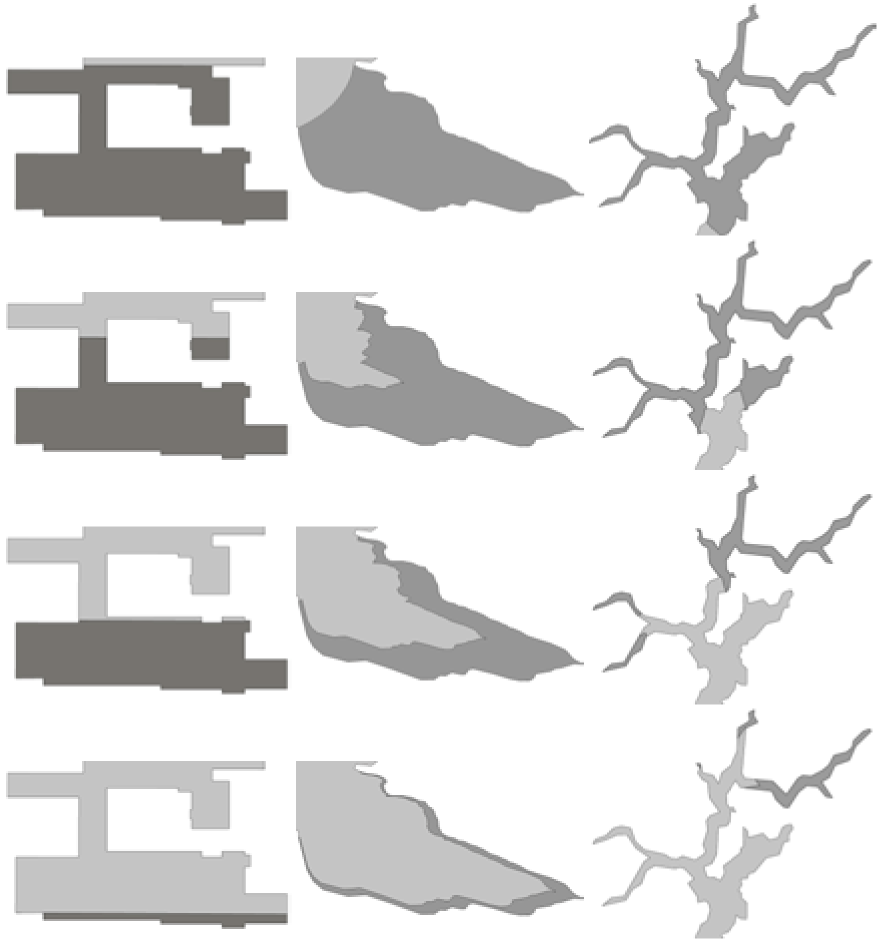
	Simple flat plane	Zipper	Eater
Always has solution	+	++	+++
Type of polygon	- (One SB only, convex, rectangular)	+ (Convex, some concave with visible SB-OB)	++ (All)
Computational efficiency	+	-	+
Multi-parts	-	+	+
Optimal shape of faces	-	++	+
Number of extra elements	+	+	-
Constant steepness	++	+	-
No vertical faces at SB	-	++	-

TABLE 6.2 Summary table, where - = bad, + = neutral and + + = good. The first three rows evaluate the algorithms in general. The remaining rows (in *italic*) give an overview of fulfilment of the mainly aesthetic soft requirements.

be minimized by an alternative graph visualization, where optimal solution lies closer to the diagonal, and therefore some options can be omitted. If there is no single part solution, the graph also gives an indication of how to split the polygon into multiple parts.

Third, the 'Eater' offers a solution for any arbitrary polygon. It always processes one triangle at the time and the whole polygon is processed in one operation; which makes the algorithm more effective and robust. The Delaunay triangulation with $\mathcal{O}(n \log n)$ is the most expensive part of the algorithm. When using the 3D structure and creating slices, the risk of multi-parts is high and it cannot guarantee constant steepness. The negative effect of multi-parts can be reduced by including more information in algorithm; *e. g.* knowledge of a triangle configuration in a previous step would lead to a better configuration for the current step. On top of that, steepness can be improved by global information about configuration of triangles. This could also improve specific cases where the loser has many holes and 'branches', and where some 'branches' are eaten faster than others.

To be able to use smooth SSC dataset generated the way mentioned above, specific types of slicing/rendering software is needed. Only then we could perform its full potential and observe the influence on the user if there any. Therefore, the next section will describe our developed prototype written as an OpenGL application supporting smooth interaction with proper user interface.



The 'Single flat plane' – example of complex building

The 'Zipper' – example of a dam

The 'Eater' – example of a river.

FIGURE 6.14 Example of three approaches with real data, the 'single flat plane' on the left, the 'Zipper' in the middle and the 'Eater' on the right. The arbitrary slides simulate gradual merge (from less merged (top) to more merged (bottom)). The light grey face is the winner, the dark grey face is the loser.

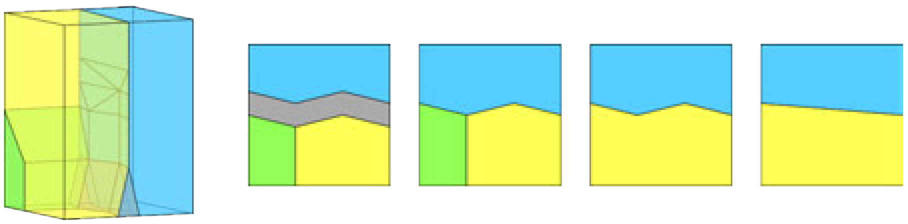


FIGURE 6.15 Example SSC data with horizontal intersections. In this example, the colours blue, grey, yellow and green respectively represent water, road, farmland and forest terrains.

§ 6.4 GPU based vario-scale viewer

As we mentioned before, the concept of map generalization research of vario-scale has shifted towards a truly smooth vario-scale structure for geographic information, see 3.4. It has the following property: A small step in the scale change leads to a small change in representation of geographic features that are represented on the map. In the implementation, the tGAP representation of 2D geo-information can be converted to a full 3D representation (smooth SSC), see Figure 3.9b on page 31.

The original SSC proposal included an explicit boundary calculation to perform the slicing with a plane. The intersection of the plane with the SSC's polyhedra yielded the set of polygons constituting the vectorial map representation. This is geometric computation intensive and not well suited for parallel execution, due to the very different workloads, depending on the polyhedras' complexity.

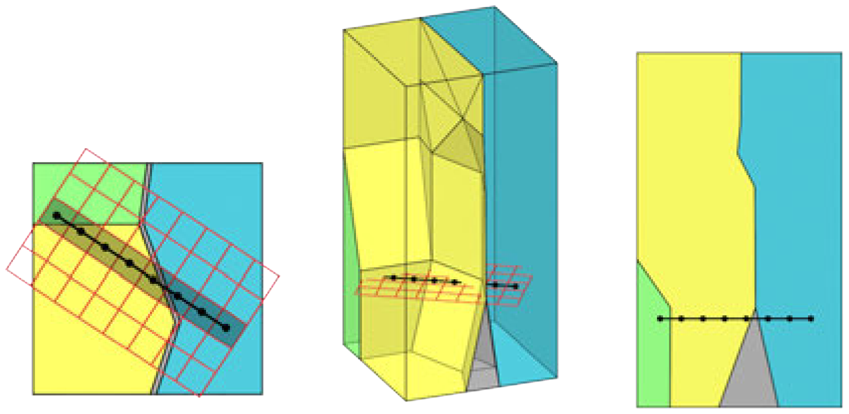
In a spirit similar to our work, some approaches (Guha et al., 2003; Hable and Rossignac, 2005) avoid calculating explicit intersections. Whenever the intersection is needed they derive only a pixel-precise location, which is sufficient. They do this by taking advantage of graphics hardware to do calculations at the level of pixels instead of primitives.

The following section describes the concept of the SSC intersection pixel rendering approach and how it is implemented on a GPU. We will use the term 'Intersector' for the implementation of the method later in this section. It describes the principle for better understanding of the environment in which the usability test would take place.

It is important to mention that this section is based on MSc thesis of Mattijs Driel, where implementation of the prototype is one of his contributions to the vario-scale project. The section covers principle of the Intersector in § 6.4.1 together with more technical details in § 6.4.2 and followed by addition rendering techniques in § 6.4.3. Even more detailed information about Intersector can be found in (Driel et al., 2016).

§ 6.4.1 Concept

Space Scale Cube representation assumes that every terrain feature represented in the data is a polyhedron (or 3D mesh) with a unique ID, and the terrain features together



(a) Raster placement viewed where the plane is intersecting.

(b) Raster placement in 3D.

(c) Side view on one row of pixels, represented as their centers.

FIGURE 6.16 The intersection method's goal is to determine what 'colour' to give each pixel in a raster. To do this, the center points of the pixels are tested for what colour polyhedron it is inside. This example highlights a single row of pixels in the raster, and determines that in this row, 1 pixel shows forest, 4 show farmland and 3 show water. Because of the low resolution and placement of the raster, the road is not visible.

form a partition of space, so each terrain feature fits tightly to its neighbours. To get a specific intersection out of the SSC, imagine a 2D pixel raster is placed somewhere inside the SSC where we want to know for each pixel what terrain feature it is intersecting. In essence, this gives us a displayable result as it is already a 2D raster. An example of this is presented in Figure 6.16.

We first explain an alternative naive intersection method, as a way to explain by contrasting it with our own. The naive method will go over all pixels and for each pixel, do a point-in-polyhedron test for all polyhedra in the data. With the degenerate case of a pixel lying on a polyhedron boundary, simply pick one of the polyhedra, the difference will hardly be visible in a full image. This method, though inefficient, does guarantee that each pixel in the raster finds the right terrain feature.

Our intersection method builds on this naive method, as it also starts with a pixel raster. We can notice how projecting each pixel along any direction will first encounter the inside boundary of the correct terrain feature: the one intersecting the pixel.

Building on this observation, we view the SSC orthogonally from the top downwards. We then render the insides of the polyhedra below the raster, while simultaneously clipping off everything above the raster.

What we essentially do here is to pick the z-axis as the projection axis, and by setting the view as orthogonal, project all pixels along the same axis. Then, rendering only below the raster will prevent any geometry above from interfering with the result. An example of this is shown in Figure 6.17.

There is one caveat in doing this method. We need to be able to assume that each pixel on the raster is in a closed polyhedron. However, because the SSC model represents a partition of space, we assume this to be the case.

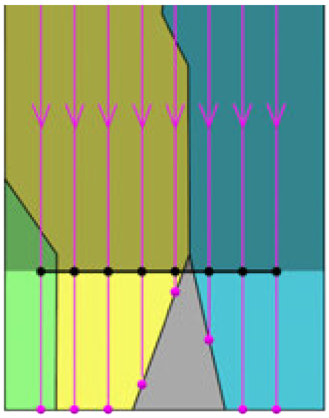


FIGURE 6.17 By projecting the points downwards we find the correct colours for the pixels. The upper shaded area must be explicitly ignored, otherwise some invalid colours will appear. In this example, if the shaded region was not clipped the most left (the first) and the fifth pixels would incorrectly get assigned yellow and blue respectively.

§ 6.4.2 Implementation

The main argument for using this raster based intersection method over one that does explicit plane-polyhedron intersections is that we only need an image as an end result. This is ideal for the GPU, specifically because it fits with the capabilities of the graphics pipeline.

Views in graphics are often a product of transforming the input vertex data into a box shaped clipping region. The clipping region represents the visible area on the x-y range, and the z-axis gives the range in which depth sorting can occur. Everything outside of the clipping region is clipped off. For the orthogonal transformation required by our intersection method, we need an orthogonal transformation matrix that specifies where in space our view is positioned.

The rendering process is initiated by feeding the GPU terrain features where the polyhedra are represented as 3D triangle meshes. Keeping with the method, we only render inside-facing triangles (method called front-face culling). Triangles are transformed from world to screen coordinates. Then pixels are computed based on orthogonal downwards viewing resulting in so-called 'fragments'. These are points that correspond to a pixel in the raster, but also maintain their post-transformation z-position used for depth sorting. We discard all fragments that appear above the z-position of the raster. We also use depth sorting on all fragments below the raster, so only the fragments closest to the raster are not discarded.

The remaining fragments store the integer ID of their corresponding terrain features into a pixel buffer. The buffer can be used to sample single IDs directly when a user wants to view information on a terrain feature under a mouse or touch cursor. Other than that, the buffer can be used as input for a full screen colour mapping technique that maps IDs to colours, which gives us our final displayable image.

§ 6.4.3 Additional rendering techniques

Our approach opens up some other possibilities, in addition to vario-scale and smooth zooming functionality, that are impossible with the traditional 2D map rendering approach: supersampling antialiasing (for sharper images), non-horizontal intersections (for mixed scale representations), near-intersection blending (for dynamic movement), and texture mapping (for enhanced map readability).

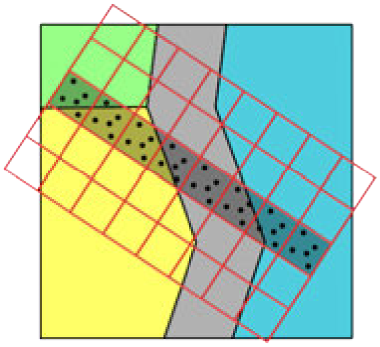


FIGURE 6.18 Using a form of SSAA, multiple samples are taken at every pixel, and their colors blended.

ANTIALIASING 2D approaches using vector based images would apply a form of antialiasing specific to vector image drawing, which is not an option for our approach. We instead use a form of supersampling antialiasing (SSAA) by re-rendering the image a few times with sub-pixel offsets and merging the result in a blending operation (Fuchs et al., 1985). Because of the static nature of maps, we can spread the re-renders out over consecutive frames to increase the image quality over time. The principle is illustrated in Figure 6.18.

NON-HORIZONTAL INTERSECTIONS As we have mentioned, the intersection method clips off geometry above the intersection plane by testing each fragment's z-position against that of the raster's plane. This implies a constant z value over the entire raster, making the intersection horizontal. We can make this more flexible by instead testing each fragment with a pre-calculated z-value that is stored in a raster sized buffer. This buffer can be prepared using any sort of function $z = f(x, y)$, implying any intersection surface that doesn't fold over in the z-axis is a valid input. Figure 6.19 shows an example using a curve shaped surface, which results in high detail near centre and lower detail near sides.

A non-horizontal intersection surface would display low and high abstraction in the same image. This could serve a function in highlighting specific areas, some examples of which are implemented in the prototype.

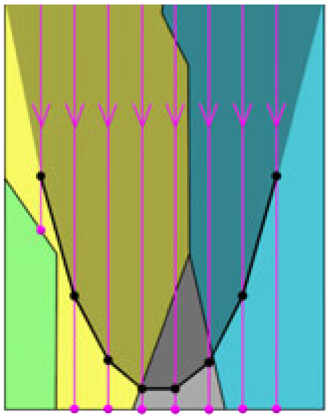


FIGURE 6.19 In this example, the center of the raster requires less abstraction than the raster edges. As a result, the forest becomes invisible and the road visible.

NEAR-INTERSECTION BLENDING When quickly zooming through much information with our approach, some changes might still feel abrupt; *e. g.* one area removed and aggregated with neighbour. Near-intersection blending provides a way to further smoothen the image by blending two terrain features that are adjacent on the z-axis. This can be conceptualized by imagining a second intersection surface placed c units below the primary intersection surface (see Figure 6.20). Any polyhedron boundaries between the two surfaces can be said to be near to the actual intersection with a distance of $d = \{0, c\}$. Below the boundary, a different terrain (and thus a different colour) is defined. If we blend the colours of both the intersecting and near-intersecting terrains with a factor of $f_{blend} = \frac{d}{c}$, we smooth the result.

In the intersection method, remember that we render the insides of terrain feature polyhedra to get the intersecting terrain. If we were to render the outsides instead, we would encounter the outside of the terrain feature directly below the intersecting terrain. Also, note that we know from the fragment what the z-position is of the boundary between the intersecting and near-intersecting terrains. We can utilize this as follows: we could do both inside and outside renders and store them. Then during colour mapping we can blend the two together with the blending factor described above.

The result is a smoother transition between terrains while actively zooming, which is visually more comfortable to look at. An example of this is shown in Figure 6.21. It does not influence anything other than the rendered colour, terrain IDs are kept intact.

TEXTURE MAPPING In the final color mapping step of the intersection method, we have used simple flat colours corresponding to the terrain feature to display for the final image. When including near-intersection blending, at most two colours are blended. Instead of, or in addition to flat colors, we can use a texture. As texture mapping characteristic we can simply use all data known to a fragment: the xyz-coordinate and the terrain feature ID.

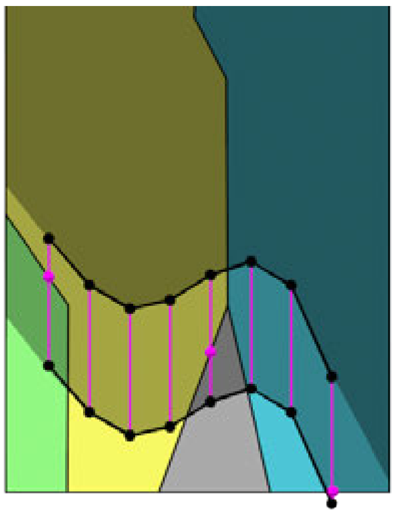


FIGURE 6.20 The three marked points represent an intersection within the given distance. The left and middle points will blend yellow-green and yellow-grey respectively. The right point intersects on the SSC boundary, so blending will not occur there.

A possible use of this is to give some types of terrain an extra visual hint to their properties (such as a tree symbol texture to a forest, or wave symbol texture in water). A simple number texture example is shown in Figure 6.22. Because the z-coordinate is also usable in the mapping, the texture can be rendered at specific ranges of abstraction levels, giving a user direct visual feedback on the level of abstraction currently visible.

§ 6.4.4 Implementation details

The Intersector prototype was written as an OpenGL ES 3.0 application in Java. Because of the use of the limited OpenGL ES feature set, the application should be possible to use on mobile phone hardware with limited efforts.

The set of intersections available for demonstration show the main classes of shapes. Shapes include a horizontal surface, a non-horizontal planar surface, a sine curve surface, a 3D mesh object based surface, and a control point surface. These shapes are shown in Figure 6.24, along with some examples of them intersecting the rural region dataset. The control point surface is based on inverse distance weighting (Shepard, 1968). This method is an example of direct real-time manipulation of the intersection shape itself.

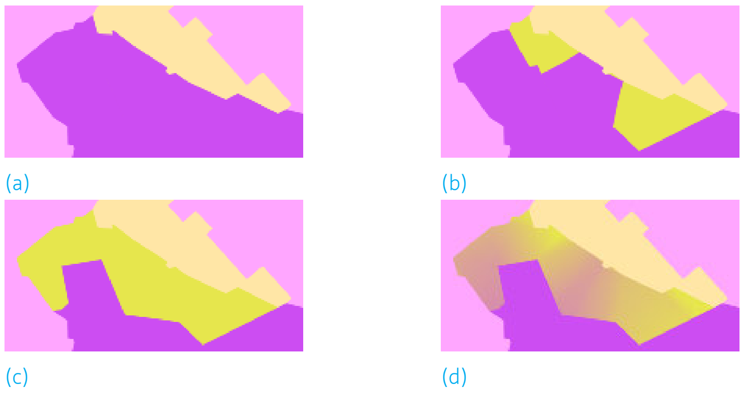


FIGURE 6.21 This example shows how a transition from one colour (a) to the next (c) can be abrupt (here shown as purple to yellow), even when the transition is non-horizontal (smoothed, or non-instant) (b). This is most noticeable while in motion, scrolling through abstractions. Blending the colours (d) can make the transition seem less abrupt.

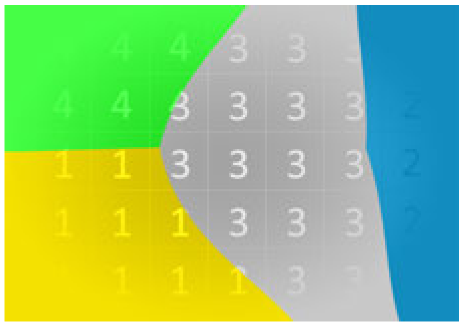


FIGURE 6.22 Texturing is shown as the numbers rendered in the regions. They fade on the sides because of the non-planar intersection used to generate the image, showing that abstraction variations influence the result.

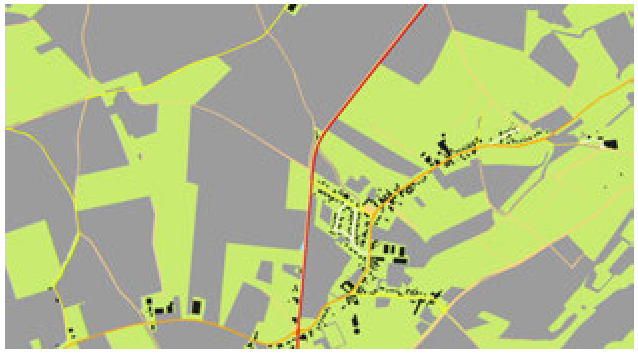
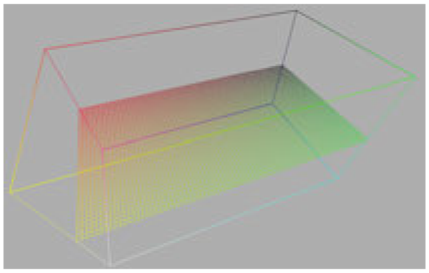
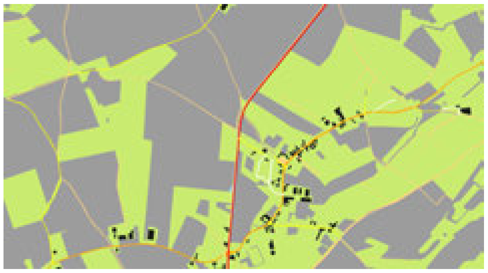


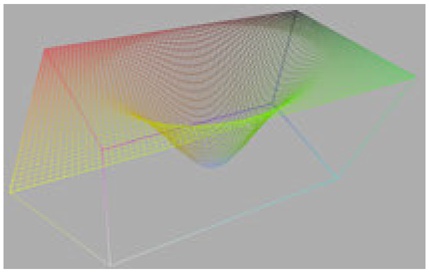
FIGURE 6.23 Subset of the testing data (source: TOP10NL, near Maastricht, NL), shown at the lowest level of abstraction.



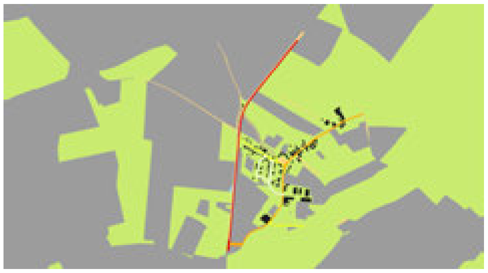
Horizontal raster.



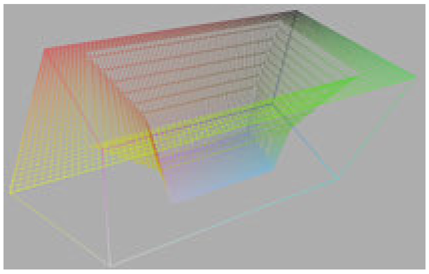
Abstraction equal everywhere.



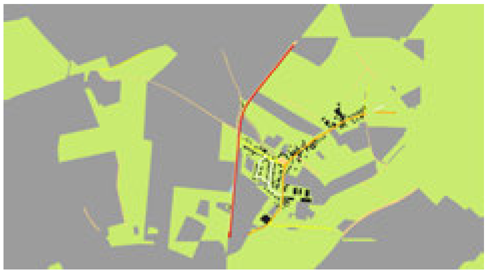
Sine curve raster.



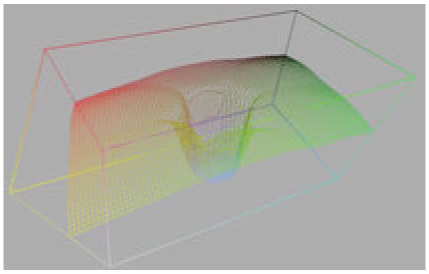
Abstraction lower in the middle.



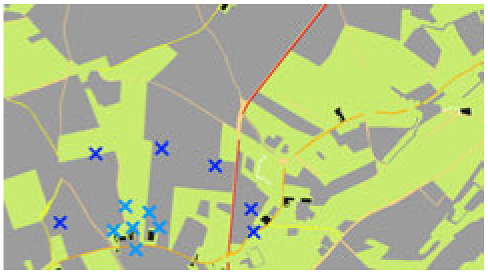
3D Mesh input raster.



Abstraction lower in the middle.



Inverse distance weighting derived raster.



Abstraction lower around the bottom left buildings, where control points (colored crosses) are placed. Dark blue and light blue crosses respectively indicate high and low abstraction. Note that the centre of the raster was shifted to the bottom left corner to generate this map.

FIGURE 6.24 Rasters on the left produce corresponding intersections shown right. With lower abstraction, more buildings and roads become visible.

§ 6.5 User's experiences

With the Intersector prototype implemented the next step was to perform a usability test with users even though it was not in our original research plan. After short round of experiments we discovered that the prototype is too complex (with more functionalities than needed) and with a non-intuitive interface. Therefore, we developed more optimized 'light' version of the Intersector and performed initial basic usability test. Here we describe initial test with preliminary results. It was carried out with a small group of users using the 'light' Intersector prototype. Only the core of our vario-scale concept was tested mainly for time reasons. For additional testing more participants, additional functionality (non-horizontal intersections, colour blending, etc.) and different SSC content with a (range of generalization techniques and options to create the structure) can be introduced. Our usability test plan included the following elements: user profiling, scope, apparatus, procedure & task and usability evaluation methods.

USER PROFILING An important initial analysis was user profiling to limit the research to a well-defined group of users. The age range of the potential user group could vary from 18 to 70 years, with no restrictions on their nationality, gender and education level. Their previous experience of map use were not important. The experiment was drawn upon the scenario of a map user observing unknown map location.

SCOPE This usability test should indicate if vario-scale maps can provide better understanding and more intuitive perception of the data compared to a fixed and limited number of maps for discrete map scales (the more traditional multi-scale approach).

The main scope was to test two types of content. The 'light' Intersector was the tool which helped us demonstrate our vario-scale approach. To eliminate the influence of the different working environment the experiment was carried out with the same 'light' Intersector prototype, that is same GUI, but for two datasets. A classical discrete multi-scale dataset (SSC content where at limited number of fixed scales the objects change: all prisms ending and succeeded by new set of prisms) and a vario-scale dataset (classic SSC based on tGAP content, see Section 3.4 on page 28), both covering the same geographic extent. We assumed that users would perform better using vario-scale maps, especially for specific tasks concerning dynamic changes of the map content such as a zooming operation. This leads us to a series of questions that the usability testing should answer:

- Can users get better understanding of data with our new vario-scale representation?
- Can users navigate faster and more effective with vario-scale maps?
- Could users also be confused by vario-scale maps and if so: how often, when and why?
- Do users benefit (more pleasant map reading experience and better quality execution of tasks) from vario-scale concept?

APPARATUS All techniques described in the previous sections were implemented as a prototype to demonstrate their potential, but the implementation was not optimized to provide the most user friendly experience, which has had a negative effect on the results of the experiment. For example, the user performs badly because he is irritated or confused by the set of mouse interactions and key strokes needed. Therefore, we implemented and used the new 'light' Intersector prototype. We reduced controls to four keys for movement and four keys for zoom in and out operation in two speeds (no mouse). Specifically, *arrow keys* for movement, *dash* (-) and *equal* (=) keys for slow, *detail* and *open bracket* ([) and *closed bracket* (]) for fast, *coarse* zooming in and out.

Note that the 'light' Intersector is implemented such a way that it can generate high rate of frames per second (FPS). Our working assumption is that 30 FPS (Read and Meyer, 2000) (in our solution can be even higher) is sufficient to provide good user impression. Smooth experience is provided by fast interaction; one key press gives immediately one new map (direct jump to target frame, throwing away some of the possibilities of our data structure). More in detail based on smooth zoom aspects, see Section 6.2 on page 88, one zoom operation is composed of one *rescaling*, one *content zoom* and one *graphic technique* (antialiasing) integrated in one new map/frame. Additionally, we offer zooming in two speeds *detail* and *coarse*, where only different is a bigger step in the scale.

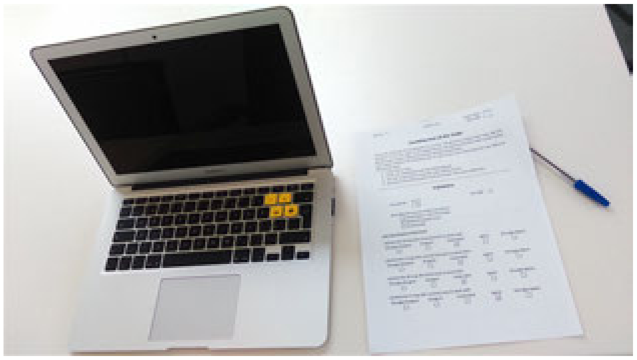


FIGURE 6.25 The experiment set-up.

For executing the experiment one notebook, powerful enough[1] to run the 'light' Intersector was used by one test person (TP) in a session of maximum 15 minutes. The 'light' Intersector ran as a window fixed in size, only showing a map view (no additional controls on screen). The equipment were set up in a quiet room to create a comfortable experiment environment where the TP accomplished the task, see Figure 6.25. The instructor was present throughout the experiment for giving an introduction and in case any unexpected situations occurs.

The test methodology consisted of a questionnaire, audio-video observation and synchronized screen recording. In addition, the screen geographic coordinates and scale denominator were recorded, with a time stamp. When the user was performing unusual actions, *e. g.* navigating very fast or extremely slow the TP was asked think aloud. All this allowed further detailed investigation of the TP's behaviour, thinking and potential hurdles during the test session.

1 MacBook Air, Early 2015

dataset	source	tGAP	Intersector	
	no. faces	no. faces	no. prisms	no. triangles
multi-scale	13.200	37.000	27.000	2.114.000
vario-scale			129.000	9.624.000

TABLE 6.3 Two datasets used in experiment. The source is topological planar partition. TGAP represents generated structure based on the source. Intersector represents generated testing datasets from tGAP where number of prisms indicates tGAP faces converted to 3D prisms.

For this experiment, we generated two main datasets; First, discrete multi-scale, represents a classic approach, where 7 discrete scales are generated based on the data factor 2 (in each dimension), *i. e.* level 0 corresponding scale denominator 1:10,000 with 13,200 features, level 1 corresponding scale denominator 1:20,000 with 3,300, level 2 corresponding 1:40,000 with 825, etc. Second, a vario-scale dataset, which captures our approach. Both were generated from the same tGAP structure. This structure has been created based on a subset of Dutch topographic map data (TOP10NL) intended for use at a 1:10,000, in regions of 9 × 9 km, see Table 6.3. Figure 6.23 shows map design used in experiment on a subset of the data. The structure has been generated with merge and split operations. Figure 6.26 shows both dataset used in the experiment in 3D view.

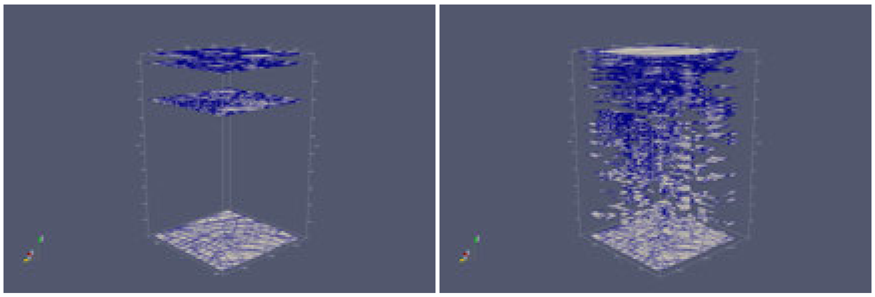

(a)Multi-scale (b)Vario-scale

FIGURE 6.26 The test datasets in 3D view.

PROCEDURE & TASK Before the experiment the TP was welcomed and the experiment was briefly introduced. Then the TP spent a few minutes randomly using the 'light' Intersector to 'get a grip' on the working environment. It was practised on similar dataset at different location of area 1 × 1 km at first. Later, we used a bigger practice dataset of area 7 × 7 km. This was done mainly to eliminate the factor of a new unfamiliar environment.

Several types of tasks based on map reading were specified: orientation, searching, analysis, routing, and planning tasks. Since we wanted perform the experiment within 15 minutes per person, it would not have been realistic to have performed all the tasks. Therefore, we chose only the following task:

I. Orientation-task: zoom out completely to see whole extent of the dataset and then try to get back as accurately as possible to the same location and zoom level as started. The time and the precision (the position of the screen) is captured.

We believed that this task had the highest chance to test our hypothesis and it can be performed multiple times in limited time to provide statistically sufficient data.

In more detail, when the experiment started, the environment was reset. The TP started with the practise dataset where controls and the task were demonstrated. When the TP felt ready we changed to the vario-scale or multi-scale dataset (half of the TPs started with the multi-scale data and other half started with vario-scale data to eliminate learning effect). Then, the TP performed the following task, as fast as the TP could three times for two different datasets ($6 \times$ total):

- TP pressed the key (*s*) to start. Intersector showed randomly generated zoomed in location.
- TP zoomed out completely to see the whole dataset.
- TP used the *arrows* keys to move the map 'off the screen'.
- TP got back to the initial location and zoom level. Then TP pressed *space* key (final position was tagged)

At the end of the session the TP had been asked filled the questionnaire to indicate their preferences. The experiment was concluded by explaining what the difference between the datasets are (the TPs did not have any understanding before) and answering additional questions.

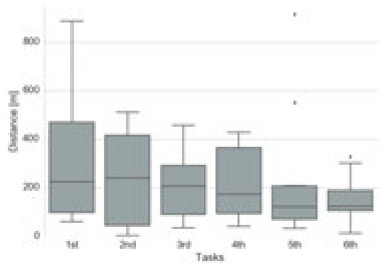
USABILITY EVALUATION METHOD The experiment was captured in questionnaires, see Appendix 7.3 on page 143. The screen geographic coordinates and scale denominator were recorded, with a time stamp. On top of that audio-video recording of both the screen and the TP took place. Furthermore, when the user was performing unusual activity, we asked the TP to talk out-loud.

The aim of the questionnaire was to get a more extended profile of the TP together with his/her opinion. It consisted of questions about the TP his/her gender, age and how often they use maps. Another part of questionnaire concerned the validation of the session, where self-reported participant ratings for satisfaction, *i. e.* ease of use of the two types of data. The TP rated the measure on a 5 point scales and it was used for quantitative measures. In last part of the questionnaire they indicated likes, dislikes, suggestions and recommendations to provide subjective measures and their preferences. This data can be then used for planning additional usability testing in the near future.

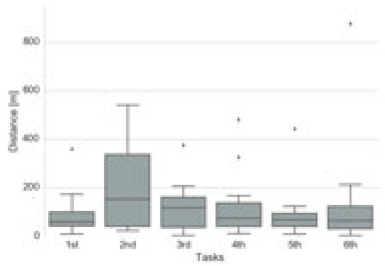
Quantitative metrics were covered by the duration of the task, the amount of time it takes the participant to complete the tasks, and accuracy of the user's performance, *i. e.* the difference in distance between real world position of starting and ending point for a given task. On top of that, the recording was intended to reveal the TP's thinking and the reasoning behind the TP's actions to give an indication where possible confusion arises.

§ 6.6 Measures

We collected data from 10 TPs in the first group and 13 TPs from the second, see more in Appendix 7.3 on page 145. We captured all tasks in the order in which they have



(a)Results after training with a small practice dataset (1×1 km) for the first group of TPs (10 persons).



(b)Results after training with a bigger practice dataset (7×7 km) for the second group of TPs (13 persons).

FIGURE 6.27 precision of the tasks in the same order as they have been presented (regardless of type of the dataset).

been performed in Figure 6.27a. A box plot graphs[2] have been used. It shows that the results are affected by the process of learning *i. e.* the users are less accurate and spend more time on the first task and they perform better later in the experiment. We assumed that this is influenced by the practice dataset, where users did not experience the environment enough. To improve this we introduced bigger practice dataset in a rural region (more similar to the testing location) of the larger area 7×7 km. Then the same experiment was run for the second group of users (13 TPs), see Figure 6.27b.

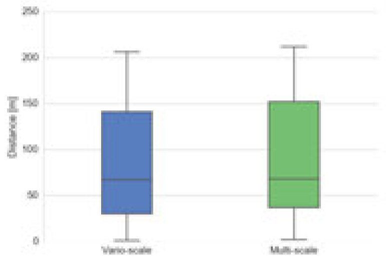
Figure 6.27b demonstrates that overall precision with the bigger practice dataset drops under 180 m in average and indicates overall more precise measurements (with significantly smaller Interquartile Ranges). It means that practise dataset had significant influence on the results. Therefore, only more precise results of the second group with 13 TPs will be considered, see Figure 6.28.

Moreover, every task is composed of three phases; (1) Zoom out, where user started at initial location and zoomed out to see the whole dataset at once. (2) Move (panning), where user used the arrow keys to shift the dataset 'out of the window' and back. (3) Zoom in, where user zoomed in to the initial location again. Figure 6.29 gives a better notion how much time is spent in which phase for the second group.

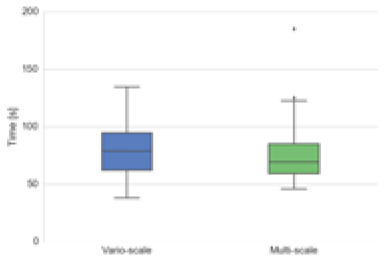
§ 6.7 Lesson learnt – smooth zoom

For initial user testing the 'light' intersector prototype has been designed in such a way that a smooth experience is provided by fast interaction; one key press gives immediately one new map. Assumption behind this was that fast response would provide the best smooth impression where the user gets new map as fast as requested. Additionally, the user's confusion is reduced because zooming in and out is less time consum-

2 The box extends from the lower to upper quartile values of the data, with a centre line at the median. Lines extending vertically from the boxes (whiskers) indicate variability outside the upper and lower quartiles. Outliers are plotted as individual points.



(a)Distance between starting and ending location



(b)Time for task

FIGURE 6.28 The results for the second group of TPs (13 persons).

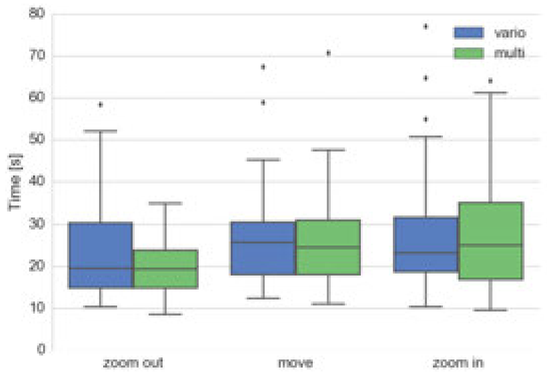


FIGURE 6.29 Three phases (zoom out, move and zoom in) of the task for vario-scale and multi-scale datasets.

ing. Operation in two speeds was provided; fast, *coarse* to quickly browse throughout the scales and slow, *detail* to perceive all small map content changes. Note that one zoom action resulting in one new map (frame) includes following smooth aspects: one *rescaling* (scale, translation), one *content zoom* (change of content) and one *graphic technique* (antialiasing).

However, the results of user testing indicates something different based on the screen recording and responses filled in the questionnaires, see Appendix on page 145. The users used only *coarse* type of interaction most of the time. Some characterised the *detail* zoom as "annoying slow zoom". This meant they could never get a smooth impression (small changes in content). The same is reflected in the responses because users could not distinguish between vario-scale (small changes in content) and multi-scale dataset. From the experiences gained here we may now define *smooth zoom* operation giving the best smooth user impression. The elements which should be included are following;

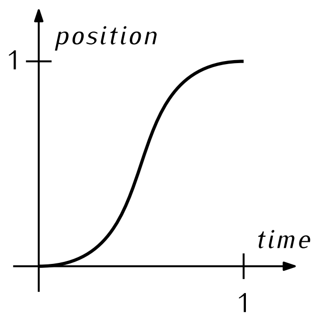
MORE STEPS FOR CONTENT ZOOM The configuration when one pressed key gives immediately one new map is not optimal. More precisely, a *content zoom* ratio of $1 : 1$ between user interaction and number of generated steps is not favourable. From initial testing it seems that every independent zoom action should return map content in more steps. This can be done by causing a *content zoom* multiple times, changing the map content only bit by bit (leading to more temporary maps, showing the transitioning of the map content in small steps from m_0 to m_1 , *e. g.* m_0, m_0', m_0'', m_0''',... , m_1), and thus progressively refining the map.

The optimal number of small steps in *content zoom* is not know. However, if we consider 30 FPS as optimal (see Apparatus on page 113), and assume one zoom action should take approximately $1/3$ of a second, which makes 10 frames for one zoom operation. Then relation $1 : n$ for *content zoom* where $n \approx 10$ is favourable.

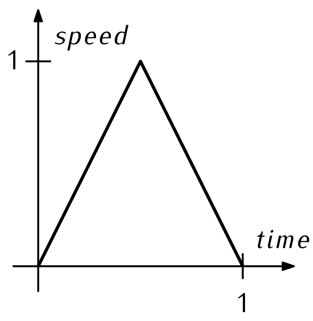
SPEED PERCEPTION The perception of the speed is another aspect that needs to be considered to provide smooth experience to the users. The illusion of speed can be simulated by the frames (temporal maps) and the timing in which they are presented, *e. g.* timing of a *content zoom* sequence m_0', m_0'', m_0''', m_1. This is similar to the process of generating intermediate frames between two images to give the appearance that the first image evolves smoothly into the second image known from animation as *inbetweening* or *tweening* (Penner, 2002).

However,if the frames are presented with long pauses between them, then the transition between consecutive maps (frames) feels stiff, artificial and mechanical. Penner (2002) claims that the notion of acceleration, a change in speed, is needed. The term *easing* is used as the transition between the states of moving and not-moving. When an object is moving, it has a velocity of some magnitude. When the object is not moving, its velocity is zero. There are many options for the shape of the acceleration. Extensive list of examples for *easing* can be found, see (Penner, 2002).

Figure 6.30 shows example of the most common easing for animation. The curve shown produces the most natural looking motion where the object speeds up from an initial still position, then slows down and stops at the end. Lifts, for example, use the same *Ease-In-Out easing* curve (Penner, 2002). We believe that similar kind of tweening together with techniques mentioned above can significantly improve the user's



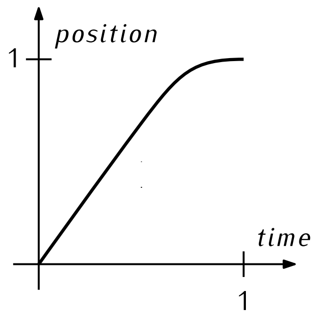
(a)Ease-In-Out easing, an example of the lift. The graph captures the situation when the object accelerates first, then slows down and comes to a stop at its destination. Source: (Penner, 2002).



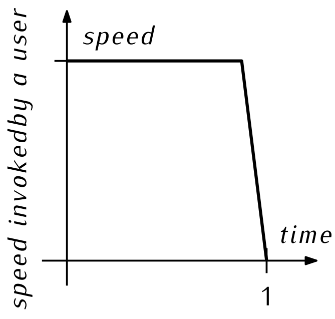
(b)The same situation as in (a). Here, the focus is on speed over the time during the Ease-In-Out easing.

FIGURE 6.30 Perception of speed by using *Ease-In-Out easing*.

overall interaction in the map. Based on our experiences so far we propose an optimal tweening for panning in Figure 6.31. Similar design can be used for zooming in and out.



(a)Our proposed tweening for map interaction. Then user's action invokes linear acceleration first slowly goes to a stop.



(b)The same situation as in (a). Here, the focus is on speed over the time during the interaction. Note that initial interaction speed depends on user's action, *i. e.* speed invokes by a user's action is equal to initial interaction speed.

FIGURE 6.31 Proposed tweening for panning which improves the user's interaction in the map.

DEPENDENT ON UI Zoom operations were provided in two speeds; fast, *coarse* and slow, *detail*. However, both operations were implemented the same way; a key press makes the map change by constant change of scale . One can imagine this as a moving the slicing plane (and scaling) in the SSC by ϵ scale where ϵ is constant, see Figure 3.9 on page 31.

The results of the testing shows that users have different preferences in zooming speed which are not really reflected in user interface (UI) *i. e.* users can only browse throughout scales by using *coarse* or *detail* zoom.

We propose to use other available UI elements supporting more levels of freedom (*e. g.* duration of action), for instance, mouse wheel or touch screen. This will makes change of scale more variable and dynamic. It will lead to the movement of the slicing plane in the SSC by ϵ where ϵ would be variable based on the duration of the action, how far it scrolled (for mouse wheel) or the size of the finger gesture (on touch screen). Then the steps for *content zoom* should be chosen and presented accordingly.

The discussion above indicates elements of smooth zoom action. An optimal use case can be described as follows: When the user performs a action such as a roll of mouse wheel or makes finger gesture a new series of content zoom steps of size ϵ of SSC is created. This series is *rescale* and is combined with *graphic techniques* to create a series of pictures/frames. Then, this sequence of frames is visualized to provide a very smooth transition from one scale to another, similar to animation – hence the term smooth zoom. Note that all this happens in a short amount of time (*e. g.* 300 ms).

§ 6.8 Reflection

In this chapter, we have tried to find indicators that the vario-scale approach can provide significant improvement in map use for users. We presented our concept of zooming together with three options how the smooth map content can be generated. Moreover, new SSC-based visualization prototype was presented. On top of that we carried out initial, preliminary usability test to indicate or verify our design assumption that vario-scale approach could provide a better user experience. This helps understand zoom action and its effect on user experience. We reflected this in Section 6.7, where also smooth zoom operation is designed.

From initial test can draw following conclusions; Figure 6.28 shows that difference between vario-scale and multi-scale in user performance. Figure 6.28a shows the vario-scale method to be more accurate ($8m$ in average). Figure 6.28b, on the other hand, presents the vario-scale as being slower ($12s$ in average). Overall, with the current number of test persons and small differences between methods we cannot reach any general conclusion. The consequence is that the initial experiment in this particular configuration is inconclusive.

We believe that the influence of other factors such as the environment, generated datasets, styling, GUI, etc. was more significant and have stronger impact on test (more than multi-scale versus simple vario-scale content). These factors cover most of the measures and, therefore, the differences between the two datasets are not obvious. Therefore, the next usability testing should included all aspects in their further developed stage. Only then further usability testing should take a place. Here the list of our hypotheses (together with our *SUGGESTIONS I* (*I* a sequence number)) why the results are not really conclusive follows:

- The majority of the TPs used only *coarse* zoom in big steps. This means that users perceived the generalized data in big steps *i. e.* They did not really experience smooth map content, but made scale jumps.
  SUGGESTION 1: Improve the zoom functionality. Users should use more convenient UI to control smooth zoom operation which includes more steps of content zoom properly integrated with rescaling and additional graphic techniques creating series of frames. These frames properly displayed would provide perception

SUGGESTION

of motion leading to better user interaction as we described in Section 6.7. It is important to keep in mind that users tend to prefer tools that they normally use, therefore, improved zoom functionality should feel "familiar".

- Since the part of the user's task is move the map 'off the screen' the users cannot keep track of their initial location. This means that they have to memorize the original position and keep it in mind ('create mental map'). Later when they see the map again they have to recall that mental map again. This makes the task more a memory exercise than a testing of the map content.
  SUGGESTION
  SUGGESTION
  SUGGESTION 2: The testing dataset should be significantly bigger.
  SUGGESTION 3: Other types of task (not memory based) should also be considered, see next bullet.
- Besides orientation task used in experiment there are other following tasks which can be consider in future:

  II. Searching-task: Locate a specific object (*e.g.* landmark/house) by zooming and panning: e.g. find church on central market square in town X.

      The time and the correct identification of the church is captured.
  III. Analysis-task 1: What is the size of central market square in square meters?

      The time and the closeness to real size of square is captured.
  IV. Routing-task 1: Find the largest lake within 500 m route via the road, starting from the church (a more nearby lake via 'the air' may be possible, but it should be checked if this can be reached via road and zoom/out in needed).

      The time and the correct identification of the lake is captured.
  V. Routing-task 2: Go to a specific house/location in a different town.

      The time and the correct identification of the house/location is captured.
  VI. Analysis-task 2: Estimate distance between two churches that are somewhat further apart.

      The time and the closeness to the real distance is captured.
  VII. Planning-task: Design a running track of about 10 km, which includes among others about 20% urban area and 30% forest tracks.

      The time and the correct quality (length and composition) of the track is assessed.
- Some faces in the structure changed their classification multiple times and then later in process changed back again. This creates impression that faces are flickering which is distracting and users have a harder time to create their mental maps, similarly to previous bullet.
  SUGGESTION
  SUGGESTION 4: The generalized dataset should be improved, methods such as grouping may help, see Section 4.6. In addition to this, the next vario-scale dataset should be a smooth-merged SSC see Section 6.3. This way, the classification change would feel more gradual.
- The map generalization is applied in such a way that buildings disappear, but no build-up area is generated. However, the buildings are one of the most significant features in the map, because they have distinct colour (black) and specific

shape. This generated an unusual situation for users, when they lost a significant orientation point.
Our recommendation will be same as *Suggestion 4*

- The datasets for the prototype contain road features. All road segments were selected from the structure, converted to triangles and stored in the prototype dataset. This method is not optimal because it increases the amount of the data drastically.

  Suggestion 5: The road features should be handled as line geometry in the prototype (used directly by GPU) instead of set of triangles as it is now. Suggestion

- We used a classic SSC model generated by the non-smooth version of merge/remove and split/collapse operations. However, some users have had difficulty to distinguish between datasets.

  Suggestion 6: The next version of a vario-scale dataset should be smooth SSC using smooth version of generalization operations namely; 1) merged/remove operation, see Section 6.3, 2) split/collapse and 3) line simplification. This way, changes of geometry would feel more gradual and the difference between vario-scale and mutli-scale datasets will be more significant. Suggestion

- So far we have developed our solution on a desktop. The next logical step should be mobile usability tests. Is should be conducted after developing a mobile implementation of the Intersector and creating other tasks for a mobile user in a real world setting. This also might include finger gestures for zoom, pan and rotate in a natural manner. Additionally it suggests a challenging solution for the 3D viewer with proper streaming on the Internet.

7 Conclusions

This thesis has researched further design and development of a system for vario-scale maps. Section 7.1 presents our ultimate vario-scale goal, which will give us perspective to understand issues addressed in the previous chapters such as further development of current generalization tools considering better vario-scale content (Chapter 4), processing of a large dataset not fitting in main memory (Chapter 5) and smooth user interaction (Chapter 6). All these aspects are now brought together in this chapter where they will be summarized and critically evaluated. The main contributions will be put in the context of the thesis by answering the research questions in Section 7.2, meanwhile new open questions and recommendation for future research will be covered in Section 7.3.

§ 7.1 Ultimate target for a vario-scale map system

Before we summarize the individual chapters, it is important to remind the reader of the ultimate target of vario-scale research. It started many years back and it developed via various improvements covered in different projects. However, the research has always been towards an optimal solution which can be characterized in the following characteristics:

I. The fully automated generalization process will provide good cartographic result comparable to NMA's products.
II. The map contains all map geometry types (points, lines, areas).
III. It will also include all type of map elements such as labels and POI.
IV. User will be able to navigate naturally throughout all scales.
V. Software will be able to process/handle an 'arbitrary' big dataset without any issues concerning cartographic quality.
VI. The generalization result will preserve topological correctness, and a full smooth 3D representation (smooth SSC cube).
VII. The solution will use well optimized progressive data transfer over the web.
VIII. Users will enjoy the smooth navigation even with limited bandwidth.
IX. The whole structure does not have to be recomputed when a small change of source data takes place.
X. It will support higher dimensions of data where, for instance, 3D data integrated with scale makes 4D Space-Scale-hyperCube.

The list above present our vision which drives the whole research line. This can be used as a context for evaluation or comparison. It also helps identify the main achievements of the PhD project.

§ 7.2 Answers to research questions

To realize technological shift in vario-scale described in previous section we raised few research sub-questions at the beginning of this thesis. Here, we provide the answers.

SUB I: *How much is the current map generalization process automated?*

Chapter 2 studied current development in the field of automated generalization. We have used output from two workshops three years apart to compare the solution for majority of European NMAs. It shown that NMAs use automated generalization in some way to provide good multi-scale representation, but it is still rapidly developing at this moment. Rarely, are there complete solutions of full automated production lines. Moreover, with current technological shift and more often budget cuts we can only assume that this tendency will continue in the future.

Other aspect is that NMAs were the main driving force so far, but this is changing. One can observe that users are starting to prefer free up-to-date data instead of cartographically correct data in longer update cycle as provided by NMAs. Moreover, in research publications significant shift to more a gradual representation is being seen, either by generating intermediate scales, morphing or continuous generalization.

This perspective shows that the research such as vario-scale is still crucial and could significantly contribute in the automated map generalization field.

SUB II: *What is the state of the art vario-scale map?*

Chapter 3 has shown the state-of-the-art prior to this PhD research. It has presented a geometrical non-redundant, variable-scale data structure in detail. The previous developments where summarized: from the GAP tree with some geometry redundancy to the topological GAP tree very well suited for a web environment where progressive refinement of vector data is supported. Additionally, it presents full 3D SSC representation (smooth tGAP), where all generalization actions lead to a smoothly changing map. A small change in the map scale results in a small geometry change.

It was shown in retrospect that the principles designed in the past are still valid and can be used for further development. Moreover, this forms the solid foundation for further development covered in this thesis.

SUB III: *How can we produce improved vario-scale data content?*

Chapter 4 focused on improving quality of content in vario-scale maps. It aimed at line features (road network) for the whole scale range, from the large, where roads are represented as areas, to mid- and small scales, where roads are represented progressively more frequently as lines. As a consequence of this process, there is an intermediate scale range where at the same time some roads are represented as areas, while others are represented as lines.

We designed a data model together with a modification of the data structure. This model is based on the integrated and explicit representation of: (1) a planar area partition; and (2) a linear road network. This enables the generalization process to include the knowledge and understanding of a linear network.

It has been presented on samples of real world data together with graphs, effectiveness analyses that automated generalization with gradual changes may lead to a reasonable results. In addition, we investigated the issue of better classification of objects during the generalization process and suggest further improvement with the groups.

SUB IV: *How can we create vario-scale data content not fitting in main memory?*

Chapter 5 shown our generic strategy to handle really huge datasets. We proposed and tested an approach to obtain a vario-scale data structure of massive datasets. We have described all steps of the process chain which we applied and tested the approach with real data.

Our solution is based on the idea of subdividing the workload according to the Fieldtree organization: a multi-level structure of space. It subdivides space regularly into fields (grid cells), at every level with a shifted origin. Only features completely fitting within a field are processed. It has been shown that our approach is generic and works for different types of input data such as land cover or road network data. We have presented two approaches for scheduling the parallel work. One open problem is formed by very large features which are only treated in the higher-levels or even top-level of the Fieldtree, i. e. postponing the actual generalization for a (too) long time.

SUB V: *How can we generate, use and validate vario-scale data to achieve a smoother impression by the user?*

Chapter 6 presented the a benefits of smooth representation, leading to a better user experience. It has also presented aspects of zoom; *rescaling*, *content zoom* and *graphic techniques*, supporting the concept of a truly smooth vario-scale structure where a delta in scale leads to a delta in the map. Then three algorithms for the merge operation to generate smooth vario-scale structure were presented in Section 6.3 where the algorithm 'eater' was shown to be the best candidate for further development, because of its generality and the fact that it always leads to a solution. On top of that to use and validate the smooth vario-scale structure we developed our visualization prototype 'Intersector'.

SUB VI: *How do the users perform with vario-scale map compare to multi-scale map?* An

initial usability test in Section 6.5 tried answer this question. We used 'light' version of our visualization prototype 'Intersector' for vario-scale data and performed an initial test for small group of participants. The result of preliminary testing was inconclusive. Therefore our hypothesis that vario-scale could be faster and better for user interaction cannot be confirmed and requires more investigation. Nevertheless, it helps us identify major obstacles and understand some factors which play significant role in user experience. Among these factors are; size and quality of generated datasets, styling, GUI and the perception of zoom action. The gained knowledge was reflected in Section 6.7, where the smooth zoom operation is discussed. We believe that if we include all elements of the experiment in their finest developed stage the question can be answer in the next usability test.

More specific aspect of vario-scale concept were addressed by sub-questions above in separate chapters. After addressing these sub-questions we have to address the main research question.

MAIN: *How to design and develop a system for vario-scale maps?*

Everything developed so far was designed in such a way that it fits in one prototype. It guarantees homogeneous solution targeting the ultimate goal presented at the beginning of this chapter, see Section 7.1. Our prototype is open source and it is shared in a repository. The code is available on following link:

`varioscale.bk.tudelft.nl`

Figure 7.1 visualises a system for vario-scale. It shows a pipeline, from a preprocessing step to the clients where data can be visualized. The architecture captures two main branches; 1) for progressive data transfer over the network which was covered in other research, 2) for smooth user experience based on SSC data. The system has been designed based on previous research. By putting individual links of the processing chain together we can see the relationships among the links and better understand our contribution (bold in Figure 7.1). We have:

- shown how the line features in our current vario-scale solution may be introduced;
- proposed a strategy provides the fully-automated generalization process that preserves a road network throughout all scales;
- investigated possibility for better classification of object with the groups during the generalization process;
- shown how to process really huge input dataset in the tGAP data structure;
- presented three algorithms for generating smooth conceptual 3D SSC model;
- develop SSC viewer (Intersector) to visualize and to experiment the data;
- performed an initial usability study;
- suggested improvements for future usability test;
- designed smooth zoom action to improve user experience.

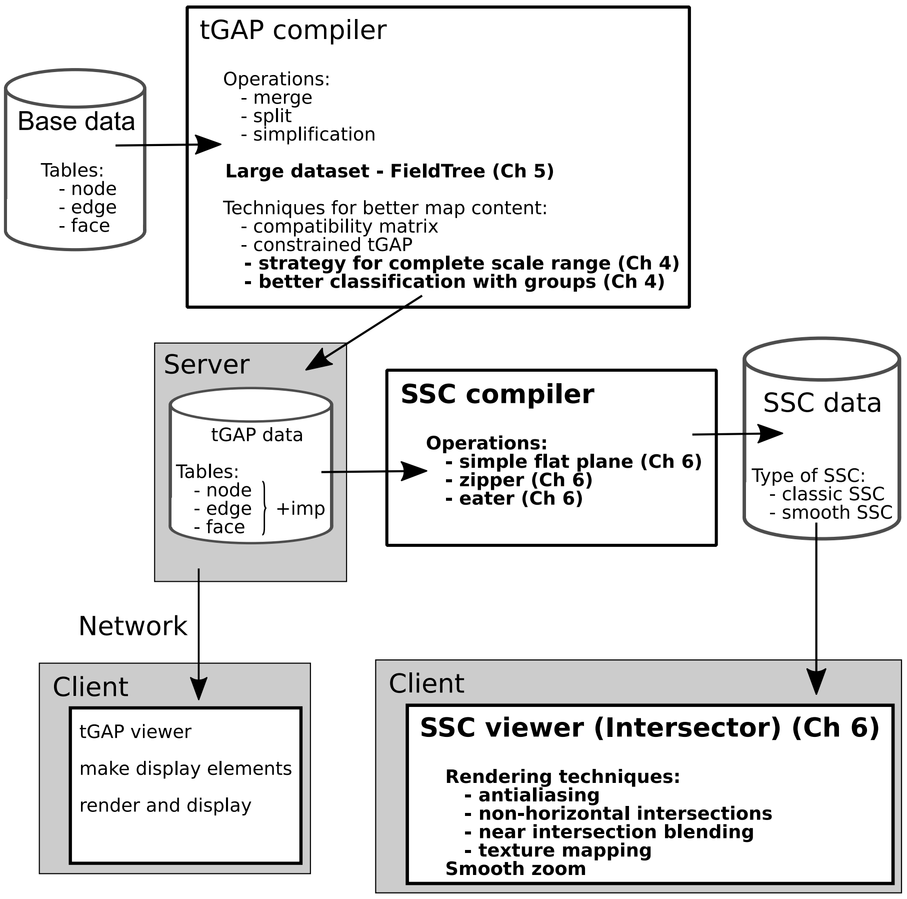


FIGURE 7.1 Diagram of a system for vario-scale maps. The contributions covered in this thesis are in **bold**.

§ 7.3 Recommendation for future work

For future research, several directions can be explored to continue with the research. Extensive list of open questions were provided in Chapter 3. Some of them were addressed in this thesis. Others are newly arising and originate from research covered in this thesis.

DATA UPDATE The current data structure is static one. It does not support any data editing. If some modification of data is made the whole structure must be recomputed. The same is true for the temporal aspect, there no version control system integrated in the structure *i. e.* going back in time is not an option. Being able to perform incremental update or any kind of data modification would be beneficial. Especial when the size of the data grows.

STYLES FOR VARIO-SCALE DATA Proper theme or colour design was not researched yet. Technical aspects and content of the structure had priority. However, aesthetic aspects of the results are important as well. Especially with more user testing in the near future. For instance, use of the same colour schema as input dataset for entire scale range does not seem correct. We should research styles for vario-scale data which do not distract users. For example, what should be the proper graphic visualization and styling of the road segment which change representation from area to line throughout the scales?

OTHER MAP FEATURES Some map features are still missing in the vario-scale concept such as Labels or POI. They are an important part of maps, but are not included in the structure. Objects in the map need a label and every label in the map takes space. It is a classical cartographic problem: each object that has to be labelled allows a number of positions where the corresponding label can be placed. However, each of these label candidates could intersect with label candidates of other objects. It is possible to find the right solution for just one predefined selected scale, but how can we find the right solution for the variable-scale?

ORCHESTRATION OF MORE TOOLS The list of operations is expanding. Our current prototype can run multiple processes at the same time. For instance, (1) algorithm dealing road network (preserving connectivity or roads), (2) processing large dataset by chunking the domain. Now, the question is how we can make possible that those two work together?

RELATION BETWEEN STEP AND SCALE The vario-scale content is produced as a sequence of steps–successively more generalized maps, so that these maps go well together instead of considering each level of generalization independently. In our perspective, the sequence of generalization steps is valuable. Our method has no explicit lower and/or upper bound with respect to the scale-range we target. This makes the method very generic. On the other hand, it is difficult to compare with current maps because it is different in principle.

DIFFICULT QUALITY ASSESSMENT Objective validation of the cartographic quality of the results is difficult. Direct comparison of the vario-scale map with existing multi-scale solutions is not adequate. Both approaches are different in principle and not really comparable. Therefore user testing together with quantitative metrics provide the only conclusive validation at this moment. New metrics for quality assessment should be researched.

IMPROVE DECISION MAKING In current selection in every process step the least important object is selected and some action is performed *e. g.* an object is collapsed or merge with its neighbour. This is done based on some condition. However, these conditions are very strict. It may not be optimal for map generalization in general, where local criteria should be in harmony with global conditions. Therefore, some more flexible or $fuzzy$ condition could provide a better solution.

WEB-SERVICES BASED ON 3D GEOMETRY Chapter 6 presents an idea that the user impression can be improved if the map is retrieved from 3D geometry (slice of the SSC prisms); First, 3D geometry is generated. Second, a slice of the geometry is calculated. Third, final maps based on the slice is constructed. In principle, it should be possible to use such a functionality in web-services setting. However, the optimal distribution of work in the web-client pipeline is not know, *e. g.* all can be performed on server side and only the final map is retrieved, or 3D geometry is sent and slicing and map construction take place on client side. There are more options possible but the optimal configuration is not known.

PROGRESSIVE DATA TRANSFER FOR 3D Furthermore, since the method is based on the data structure optimized for progressive transfer as has been shown for 2D vector data in (Huang et al., 2016), the same could be done for 3D data in theory. How can we keep data transfer at a minimum also for explicit 3D storage, and is explicit 3D storage really needed? Storing the data in a 3D topology data structure could be one option. Another option could be to derive what is needed from the tGAP data structures that store more or less separately the 2D geometry and 1D scale range and create the 3D representation when needed, *e. g.* at client side for slicing and map constructing.

HIGHER DIMENSIONALITY OF VARIO-SCALE DATA Higher dimensionality of vario-scale data is not explored. When we integrate 3D SSC with scale leading to integrated 4D Space-Scale-hyperCube data. Then the intersection of this 4D hypercube with the hyperplane gives a perfect 3D topology (no gaps or overlaps). This could solve a big problem as is often the case in the transition from one Level of Detail (LOD) to the next LOD in computer graphics.

MORE CONCLUSIVE USABILITY TESTING Initial user testing was done with small groups of participants and differences in the results were minor, and therefore can be considered as inconclusive. However, it was only a starting point in the research into user perception of vario-scale. We still believe that with proper map, environment, styling and GUI significant results can be achieved in near future. In long term, more testing should follow such as different generated SSC cubes, alternative visualization technique with various intersection shapes or different zooming techniques. All these have an effect on users and only basic options have been tested so far.

Bibliography

Ai, T. and van Oosterom, P. (2002). Gap-tree extensions based on skeletons. In *Advances in Spatial Data Handling, 10th International Symposium on Spatial Data Handling, Ottawa, Canada*. (Cited on pages 5, 21, and 22).

Armstrong, M. P. and Densham, P. J. (1992). Domain decomposition for parallel processing of spatial problems. *Computers, Environment and Urban Systems*, 16(6):497 – 513. (Cited on page 73).

Ballard, D. H. (1981). Strip trees: A hierarchical representation for curves. *Commun. ACM*, 24(5):310–321. (Cited on page 20).

Bereuter, P., Weibel, R., and Burghardt, D. (2012). Content zooming and exploration for mobile maps. In Gensel, J., Josselin, D., and Vandenbroucke, D., editors, *Multidisciplinary Research on Geographical Information in Europe and Beyond Proceedings of the AGILE'2012 International Conference on Geographic Information Science, Avignon, April, 24-27, 2012*, pages 74–80. International Conference on Geographic Information Science, Avignon. (Cited on page 18).

Bertolotto, M. and Egenhofer, M. J. (2001). Progressive transmission of vector map data over the world wide web. *GeoInformatica*, 5(4):345–373. (Cited on page 20).

Bregt, A. and Bulens, J. (1996). Application-oriented generalization of area object. In Molenaar, M., editor, *Methods for The Generalization of Geo-databases*, number 43, pages 57–64. Netherlands Geodetic Commission, Delft, The Netherlands. (Cited on page 22).

Brewer, C. A. and Buttenfield, B. P. (2007). Framing guidelines for multi-scale map design using databases at multiple resolutions. *Cartography and Geographic Information Science*, 34(1):3–15. (Cited on page 15).

Buttenfield, B. P. (2002). *Transmitting Vector Geospatial Data across the Internet*, pages 51–64. Springer Berlin Heidelberg, Berlin, Heidelberg. (Cited on page 20).

Buttenfield, B. P. and McMaster, R. B. (1991). *Map Generalization : Making Rules for Knowledge Representation*. Longman Scientific & Technical, Harlow, UK. (Cited on page 14).

Cecconi, A. (2003). *Integration of cartographic generalization and multi-scale databases for enhanced web mapping*. PhD thesis, University of Zurich. (Cited on page 15).

Cecconi, A., Weibel, R., and Barrault, M. (2002). Improving automated generalisation for on-demand web mapping by multiscale databases. In *Advances in Spatial Data Handling*, pages 515–531. Springer Berlin Heidelberg. (Cited on page 14).

Chen, J., Hu, Y., Li, Z., Zhao, R., and Meng, L. (2009). Selective omission of road features based on mesh density for automatic map generalization. *International Journal of Geographical Information Science*, 23(8):1013–1032. (Cited on page 42).

Chimani, M., van Dijk, T. C., and Haunert, J.-H. (2014). How to eat a graph: Computing selection sequences for the continuous generalization of road networks. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 243–252, New York, NY, USA. ACM. (Cited on pages 15, 37, 40, and 88).

Danciger, J., Devadoss, S. L., Mugno, J., Sheehy, D., and Ward, R. (2009). Shape deformation in continuous map generalization. *GeoInformatica*, 13(2):203–221. (Cited on pages 15, 16, and 88).

Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA. USENIX Association. (Cited on page 73).

Dilo, A., van Oosterom, P., and Hofman, A. (2009). Constrained tgap for generalisation between scales: the case of dutch topographic data. *Computers, Environment and Urban Systems*, 33(5):388–402. (Cited on pages 5, 26, 28, and 63).

Ding, Y. and Densham, P. J. (1996). Spatial strategies for parallel spatial modelling. *International journal of geographical information systems*, 10(6):669–698. (Cited on page 73).

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. (Cited on page 21).

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122. (Cited on page 20).

Driel, M., Šuba, R., Meijers, M., van Oosterom, P., and Eisemann, E. (2016). Real-time space-scale cube slicing for interactive geovisualization. *to be submitted in Computers & Graphics*. (Cited on page 105).

Duchêne, C., Baella, B., Brewer, C. A., Burghardt, D., Buttenfield, B. P., Gaffuri, J., Käuferle, D., Lecordix, F., Maugeais, E., Nijhuis, R., Pla, M., Post, M., Regnauld, N., Stanislawski, L. V., Stoter, J., Tóth, K., Urbanke, S., van Altena, V., and Wiedemann, A. (2014). *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, volume 2014 of *Lecture Notes in Geoinformation and Cartography*, chapter Generalization in Practice Within National Mapping Agencies, pages 329–391. Springer International Publishing. (Cited on pages 12, 13, and 14).

Dumont, M., Touya, G., and Duchêne, C. (2015). Automated generalisation of intermediate levels in a multi-scale pyramid. In *Proceedings of 18th ICA Workshop on Generalisation and Multiple Representation*, Rio de Janeiro, Brazil. (Cited on page 18).

Dumont, M., Touya, G., and Duchêne, C. (2016). Assessing the variation of visual complexity in multi-scale maps with clutter measures. In *Proceedings of 19th ICA Workshop on Generalisation and Multiple Representation*, Helsinki, Finland. (Cited on page 18).

Eastman, C. M. and Weiler, K. J. (1979). *Geometric Modeling Using the Euler Operators*. DRC. Institute of Physical Planning, Carnegie-Mellon University. (Cited on page 15).

Edwardes, A. and Mackaness, W. (2000). Intelligent road network simplification in urban areas. *GIS Research UK 2000 Conference (GISRUK 2000), University of York*. (Cited on page 42).

Eldawy, A., Li, Y., Mokbel, M. F., and Janardan, R. (2013). Cg_hadoop: Computational geometry in mapreduce. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 294–303, New York, NY, USA. ACM. (Cited on page 73).

Foerster, T., Stoter, J., and Kraak, M.-J. (2010). Challenges for automated generalisation at european mapping agencies: a qualitative and quantitative analysis. *The Cartographic Journal*, 47(1):41–54. (Cited on pages 12 and 13).

Frank, A. U. and Barrera, R. (1990). The Fieldtree: A data structure for Geographic Information Systems. In *Proceedings of the First Symposium on Design and Implementation of Large Spatial Databases*, SSD '90, pages 29–44, New York, NY, USA. Springer-Verlag New York, Inc. (Cited on pages 72 and 74).

Fuchs, H., Goldfeather, J., Hultquist, J. P., Spach, S., Austin, J. D., Brooks, Jr., F. P., Eyles, J. G., and Poulton, J. (1985). Fast spheres, shadows, textures, transparencies, and image enhancements in pixel-planes. *SIGGRAPH Comput. Graph.*, 19(3):111–120. (Cited on page 108).

Guan, X., Wu, H., and Li, L. (2012). A Parallel Framework for Processing Massive Spatial Data with a Split-and-Merge Paradigm. *Transactions in GIS*, 16(6):829–843. (Cited on page 73).

Guha, S., Krishnan, S., Munagala, K., and Venkatasubramanian, S. (2003). Application of the two-sided depth test to CSG rendering. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 177–180. ACM. (Cited on page 105).

Günther, O. (1988). *Efficient Structures for Geometric Data Management*. Springer-Verlag New York, Inc., New York, NY, USA. (Cited on page 20).

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57. (Cited on page 21).

Hable, J. and Rossignac, J. (2005). Blister: GPU-based rendering of boolean combinations of free-form triangulated shapes. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1024–1031. ACM. (Cited on page 105).

Hägerstrand, T. (1970). What about people in regional science? *Papers of the Regional Science Association*, 24(1):6–21. (Cited on page 30).

Hampe, M., Sester, M., and Harrie, L. (2004). Multiple representation databases to support visualization on mobile devices. In *Proceedings of the XXth ISPRS congress, July 12-23, 2004, Istanbul, Turkey, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXV(B4:IV)*, pages 135–140. (Cited on page 32).

Harrie, L., Sarjakoski, L. T., and Lehto, L. (2002). A variable-scale map for small-display cartography. In *Joint International Symposium on GeoSpatial Theory, Processing and Applications (ISPRS/Commission IV, SDH2002)*, Ottawa, Canada. (Cited on page 32).

Harrie, L., Stigmar, H., and Djordjevic, M. (2015). Analytical estimation of map readability. *ISPRS International Journal of Geo-Information*, 4(2):418. (Cited on page 18).

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. (Cited on pages 26, 28, and 63).

Haunert, J.-H. and Wolff, A. (2006). Generalization of land cover maps by mixed integer programming. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '06, pages 75–82, New York, NY, USA. ACM. (Cited on page 63).

Hawick, K. A., Coddington, P. D., and James, H. A. (2003). Distributed frameworks and parallel algorithms for processing large-scale geographic data. *Parallel Computing*, 29(10):1297–1333. (Cited on page 73).

Hildebrandt, J., Owen, M., and Hollamby, R. (2010). Cluster raptor: Dynamic geospatial imagery visualisation using backend repositories. In *Proceedings of the 5th International Command and Control Research and Technology Symposium (ICCRTS)*. (Cited on page 20).

Huang, L., Meijers, M., Šuba, R., and van Oosterom, P. (2016). Engineering web maps with gradual content zoom based on streaming vector data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:274 – 293. (Cited on pages 5, 8, 19, 34, 35, 36, 40, 87, and 131).

ICA and EuroSDR (2013). ICA/EuroSDR NMA Symposium 2013. Accessed 17 June 2013. (Cited on pages 12 and 13).

ICA and EuroSDR (2015). 2nd ICA/EuroSDR NMA Symposium. Accessed 29 June 2016. (Cited on pages 12 and 13).

Jones, C. B., Abdelmoty, A. I., Lonergan, M. E., van der Poorten, P., and Zhou, S. (2000). Multi-scale spatial database design for online generalisation. In *9th International Symposium on Spatial Data Handling, Beijing, International Geographical Union*, volume 7b, pages 34–44. (Cited on page 20).

Jones, C. B. and Abraham, I. M. (1987). Line generalisation in a global cartographic database. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 24(3):32–45. (Cited on page 20).

Käuferle, D. (2015). New national maps for switzerland. Presented at 2nd ICA/EuroSDR NMA Symposium 2015, Amsterdam, The Netherlands. Accessed 10 August 2016. (Cited on page 12).

Lagrange, J.-P., Weibel, R., and Muller, J.-C. (1995). *GIS and Generalisation: Methodology and Practice*. Number 1. CRC Press. (Cited on page 14).

Lazaridis, I. and Mehrotra, S. (2001). Progressive approximate aggregate queries with a multi-resolution tree structure. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 401–412, New York, NY, USA. ACM. (Cited on page 20).

Li, Z. and Zhou, Q. (2012). Integration of linear and areal hierarchies for continuous multi-scale representation of road networks. *International Journal of Geographical Information Science*, 26(5):855–880. (Cited on page 42).

MacEachren, A. M. and Kraak, M.-J. (1997). Exploratory cartographic visualization: advancing the agenda. *Computers & Geosciences*, 23(4):335–343. (Cited on page 14).

Mackaness, W., Burghardt, D., and Duchêne, C. (2014). *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, volume 2014 of *Lecture Notes in Geoinformation and Cartography*, chapter Map generalisation: Fundamental to the Modelling and Understanding of Geographic Space, pages 1–14. Springer International Publishing. (Cited on pages 9, 10, and 11).

Mackaness, W. A., Ruas, A., and Sarjakoski, L. T. (2007). *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier, Oxford, UK. (Cited on page 14).

Meijers, M. (2011). *Variable-scale Geo-information*. PhD thesis, Delft University of Technology. (Cited on pages 3, 5, 26, 27, 29, 30, 31, 32, 41, and 46).

Meijers, M. and Ledoux, H. (2013). EdgeCrack: a parallel Divide-and-Conquer algorithm for building a topological data structure. In Ellul, C., Zlatanova, S., Rumor, M., and Laurini, R., editors, *Urban and Regional Data Management: Annual 2013*, pages 107–116. CRC Press. (Cited on page 73).

Meijers, M., Savino, S., and van Oosterom, P. (2016). SPLITAREA : an algorithm for weighted splitting of faces in the context of a planar partition. *International Journal of Geographical Information Science*, 30:1522–1551. (Cited on page 53).

Meijers, M. and van Oosterom, P. (2011). The space-scale cube: An integrated model for 2d polygonal areas and scale. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(4):95–102. (Cited on pages 3 and 29).

Meijers, M., van Oosterom, P., and Quak, W. (2009). A storage and transfer efficient data structure for variable scale vector data. In Sester, M., Bernard, L., and Paelke, V., editors, *Advances in GIScience*, Lecture Notes in Geoinformation and Cartography, pages 345–367. Springer Berlin Heidelberg. (Cited on pages 23 and 26).

Meijers, M., Šuba, R., and van Oosterom, P. (2015). Parallel creation of vario-scale data structures for large datasets. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-4/W7:1–9. (Cited on pages 8 and 71).

Meng, L., Huang, C., Zhao, C., and Lin, Z. (2007). An improved Hilbert curve for parallel spatial data partitioning. *Geo-spatial Information Science*, 10(4):282–286. (Cited on page 73).

Midtbø, T. and Nordvik, T. (2007). Effects of animations in zooming and panning operations on web maps: A web-based experiment. *The Cartographic Journal*, 44(4):292–303. (Cited on pages 40 and 88).

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. (Cited on pages 15 and 88).

Penner, R. (2002). *Robert Penner's Programming Macromedia Flash MX*, chapter 7: Motion, Tweening, and Easing, pages 191–240. McGraw-Hill, Inc., New York, NY, USA, 1 edition. (Cited on pages 119 and 120).

Read, P. and Meyer, M.-P. (2000). *Restoration of Motion Picture Film (Butterworth-Heinemann Series in Conservation and Museology)*. Butterworth-Heinemann. (Cited on page 114).

Reilly, D. F. and Inkpen, K. M. (2004). Map morphing: Making sense of incongruent maps. In *Proceedings of Graphics Interface 2004*, GI '04, pages 231–238, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society. (Cited on page 88).

Reilly, D. F. and Inkpen, K. M. (2007). White rooms and morphing don't mix: setting and the evaluation of visualization techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 111–120, New York, NY, USA. ACM. (Cited on page 88).

Robinson, A. H., Morrison, J. L., Muehrcke, P. C., Kimerling, A. J., and Guptill, S. C. (1995). *Elements of cartography*, chapter 29. Dynamic/interactive mapping. Wiley, Wiley, New York, 6th edition. (Cited on page 14).

Rosenbaum, R. and Schumann, H. (2004). Remote raster image browsing based on fast content reduction for mobile environments. In *Proceedings of the Seventh Eurographics Conference on Multimedia*, EGMM'04, pages 13–19, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. (Cited on page 20).

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260. (Cited on page 20).

Sester, M. and Brenner, C. (2005). Continuous generalization for visualization on small mobile devices. In *Developments in Spatial Data Handling*, pages 355–368. Springer Berlin Heidelberg. (Cited on pages 15, 40, and 88).

Sharker, M. H. and Karimi, H. A. (2014). *Big Data*, chapter Distributed and Parallel Computing, pages 1–30. CRC Press. (Cited on page 72).

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA. ACM. (Cited on page 110).

Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Lin, M. C. and Manocha, D., editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag. From the First ACM Workshop on Applied Computational Geometry. (Cited on page 97).

Stoter, J. (2005). Generalisation within NMA's in the 21st century. In *Proceedings of the International Cartographic Conference*. (Cited on page 12).

Stoter, J., Meijers, M., van Oosterom, P., Grunreich, D., and Kraak, M.-J. (2010). Applying dlm and dcm concepts in a multi-scale data environment. In *GDI 2010, a Symposium on Generalization and Data Integration, Boulder*, page 7. (Cited on page 4).

Stoter, J., Post, M., van Altena, V., Nijhuis, R., and Bruns, B. (2014). Fully automated generalization of a 1:50k map from 1:10k data. *Cartography and Geographic Information Science*, 41(1):1–13. (Cited on pages 17 and 73).

Stoter, J., van Altena, V., Post, M., Burghardt, D., and Duchêne, C. (2016). Automated generalisation within NMAs in 2016. In *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLI-B4, 647-652, 2016*, Prague. (Cited on pages 12, 13, and 14).

Thiemann, F., Warneke, H., Sester, M., and Lipeck, U. (2011). A Scalable Approach for Generalization of Land Cover Data. In Geertman, S., Reinhardt, W., and Toppen, F., editors, *Advancing Geoinformation Science for a Changing World*, Lecture Notes in Geoinformation and Cartography, pages 399–420. Springer Berlin Heidelberg. (Cited on page 73).

Thomson, R. C. and Richardson, D. E. (1999). The 'good continuation' principle of perceptual organization applied to the generalization of road networks. In *Proceedings of the 19th International Cartographic Conference*, pages 1215–1225. Ottwawa. (Cited on page 42).

Touya, G. and Girres, J.-F. (2013). Scalemaster 2.0: a scalemaster extension to monitor automatic multi-scales generalizations. *Cartography and Geographic Information Science*, 40(3):192–200. (Cited on page 15).

Turner, A. (2007). From axial to road-centre lines: a new representation for space syntax and a new model of route choice for transport network analysis. *Environment and Planning B: Planning and Design*, 34(3):539–555. (Cited on pages 42 and 47).

Tutorials Point (2015). Software development life cycle. In *Software Engineering Tutorial*. Accessed 8 August 2016. (Cited on page 6).

Uitermark, H., Vogels, A., and van Oosterom, P. (1999). Semantic and geometric aspects of integrating road networks. In *2nd International conference on Interoperating Geographic Information Systems, 10-12 March'99, Zurich, LNCS 1580*, pages 177–188. Springer-Verlag. (Cited on page 47).

van Kreveld, M. (2001). Smooth generalization for continuous zooming. In *Proceedings 20th International Cartographic Conference (ICC'01)*, pages 2180–2185, Beijing, China. (Cited on page 14).

van Oosterom, P. (1989). Reactive data structures for geographic information systems. In *AutoCarto 9, Baltimore, Maryland, 1986*, pages 665–674. (Cited on page 20).

van Oosterom, P. (1990). *Reactive Data Structures for Geographic Information Systems*. PhD thesis, Department of Computer Science, Leiden University, The Netherlands. (Cited on pages 20 and 95).

van Oosterom, P. (1992). A storage structure for a multi-scale database: The reactive-tree. 16(3):239–247. (Cited on page 20).

van Oosterom, P. (1994). *Reactive Data Structures for Geographic Information Systems*. Oxford University Press, Inc., New York, NY, USA. (Cited on page 21).

van Oosterom, P. (1995). *GIS and Generalization, Methodology and Practice*, chapter The GAP-tree, an approach to "On-the-Fly" Map Generalization of an Area Partitioning., pages 120–132. (Cited on pages 21 and 22).

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(11):331–346. (Cited on pages 2, 5, 23, 24, 25, 28, 33, and 41).

van Oosterom, P. and Meijers, M. (2011a). Method and system for generating maps in an n-dimensional space. Dutch patent application 2006630, WO 2012/144983, filed April 19, 2011, published October 26, 2012. (Cited on page 5).

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. (Cited on pages 2 and 90).

van Oosterom, P. and Meijers, M. (2012). Vario-scale data structures supporting smooth zoom and progressive transfer of 2d and 3d data. In *Jaarverslag 2011*, pages 21–42. Nederlandse Commissie voor Geodesie. (Cited on page 33).

van Oosterom, P., Meijers, M., Stoter, J., and Šuba, R. (2014). *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, volume 2014 of *Lecture Notes in Geoinformation and Cartography*, chapter Data Structures for Continuous Generalisation: tGAP and SSC, pages 83–118. Springer International Publishing. (Cited on pages 2, 8, and 19).

van Oosterom, P. and Schenkelaars, V. (1995). The development of an interactive multi-scale GIS. *International Journal of Geographical Information Systems*, 9(5):489–507. (Cited on page 21).

van Oosterom, P. and Vijlbrief, T. (1996). The Spatial Location Code. In *Proceedings of the 7th International Symposium on Spatial Data Handling*, SDH'96. (Cited on page 74).

van Putten, J. and van Oosterom, P. (1998). New results with Generalized Area Partitionings. In *Proceedings of the International Symposium on Spatial Data Handling*, SDH'98, pages 485–495. (Cited on pages 21, 22, 63, and 72).

van Smaalen, J. W. N. (2003). *Automated aggregation of geographic objects. A new approach to the conceptual generalisation of geographic databases*. PhD thesis, Wageningen University, The Netherlands. (Cited on page 22).

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. *7th International conference on GeoComputation*. (Cited on pages 21, 23, and 29).

Visvalingam, M. and Whyatt, J. D. (1993). Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1):46–51. (Cited on page 91).

Šuba, R. (2012). Content of variable-scale maps. Number 64, pages 1–36. ISBN: 978-90-77029-38-1 ISSN: 1569-0245. (Cited on page 33).

Šuba, R., Driel, M., Meijers, M., van Oosterom, P., and Eisemann, E. (2016a). Usability test plan for truly vario-scale maps. In *Proceedings of the 19th ICA Workshop on Generalisation and Multiple Representation, Helsinki, Finland*. (Cited on pages 8 and 87).

Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014a). An area merge operation for smooth zooming. In Huerta, J., Schade, S., and Granell, C., editors, *Connecting a Digital Europe Through Location and Place*, Springer Lecture Notes in Geoinformation and Cartography, pages 275–293. Springer International Publishing. ISBN: 978-3-319-03611-3. (Cited on pages 8, 19, and 87).

Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014b). Continuous Road Network Generalisation. In *Proceedings of the 17th ICA Workshop on Generalisation and Multiple Representation, Vienna, Austria, September 23, 2014*, pages 1–12. (Cited on pages 8, 39, 52, and 78).

Šuba, R., Meijers, M., and Oosterom, P. v. (2016b). Continuous road network generalization throughout all scales. *ISPRS International Journal of Geo-Information*, 5(8):145. (Cited on pages 8 and 39).

Šuba, R., Meijers, M., and van Oosterom, P. (2013). 2D vario-scale representations based on real 3D structure. In *Proceedings of the 16th ICA Generalization Workshop*, pages 1–11. The paper for ICA worshop 2013/Dresden. (Cited on pages 8 and 102).

Šuba, R., Meijers, M., and van Oosterom, P. (2015). Large scale road network generalization for vario-scale map. In *Proceedings of the 18th ICA Workshop on Generalisation and Multiple Representation, Rio de Janeiro, Brazil, 21 August, 2015*, pages 1–10. (Cited on pages 8, 39, and 43).

Wang, H., Fu, X., Wang, G., Li, T., and Gao, J. (2011). A common parallel computing framework for modeling hydrological processes of river basins. *Parallel Computing*, 37(6–7):302 – 315. (Cited on page 73).

Weibel, R. (1997). *Generalization of spatial data: Principles and selected algorithms*, volume 1340 of *Algorithmic Foundations of Geographic Information Systems*. Springer, Berlin, Germany, Berlin, Heidelberg. (Cited on page 14).

Weiss, R. and Weibel, R. (2014). Road network selection for small-scale maps using an improved centrality-based algorithm. *Journal of Spatial Information Science*, 2014(9):71–99. (Cited on page 41).

Zhang, X. (2012). *Automated evaluation of generalized topographic maps*. PhD thesis, University of Twente. (Cited on pages 63 and 66).

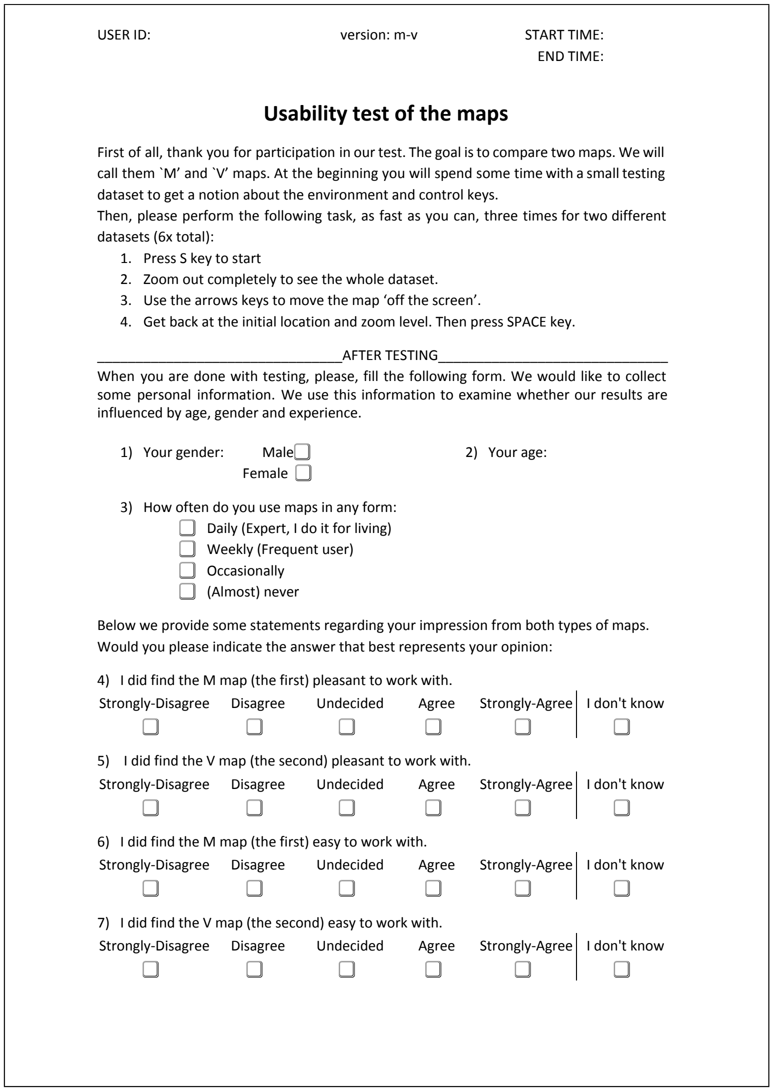
Zhou, Q. and Li, Z. (2012). A comparative study of various strategies to concatenate road segments into strokes for map generalization. *International Journal of Geographical Information Science*, 26(4):691–715. (Cited on pages 42 and 48).

Zhou, X., Abel, D., and Truffet, D. (1998). Data Partitioning for Parallel Spatial Join Processing. *GeoInformatica*, 2(2):175–204. (Cited on page 73).

Zhou, X., Prasher, S., Sun, S., and Xu, K. (2004). Multiresolution spatial databases: Making web-based spatial applications faster. In *Advanced Web Technologies and Applications: 6th Asia-Pacific Web Conference, APWeb 2004, Hangzhou, China, April 14-17, 2004. Proceedings*, pages 36–47, Berlin, Heidelberg. Springer Berlin Heidelberg. (Cited on page 20).

Appendix

Questionnaire



USER ID: version: m-v START TIME:
END TIME:

Usability test of the maps

First of all, thank you for participation in our test. The goal is to compare two maps. We will call them `M' and `V' maps. At the beginning you will spend some time with a small testing dataset to get a notion about the environment and control keys.
Then, please perform the following task, as fast as you can, three times for two different datasets (6x total):

1. Press S key to start
2. Zoom out completely to see the whole dataset.
3. Use the arrows keys to move the map 'off the screen'.
4. Get back at the initial location and zoom level. Then press SPACE key.

______________________________AFTER TESTING______________________________
When you are done with testing, please, fill the following form. We would like to collect some personal information. We use this information to examine whether our results are influenced by age, gender and experience.

1) Your gender: Male ☐ Female ☐ 2) Your age:

3) How often do you use maps in any form:
- ☐ Daily (Expert, I do it for living)
- ☐ Weekly (Frequent user)
- ☐ Occasionally
- ☐ (Almost) never

Below we provide some statements regarding your impression from both types of maps. Would you please indicate the answer that best represents your opinion:

4) I did find the M map (the first) pleasant to work with.

Strongly-Disagree	Disagree	Undecided	Agree	Strongly-Agree	I don't know
☐	☐	☐	☐	☐	☐

5) I did find the V map (the second) pleasant to work with.

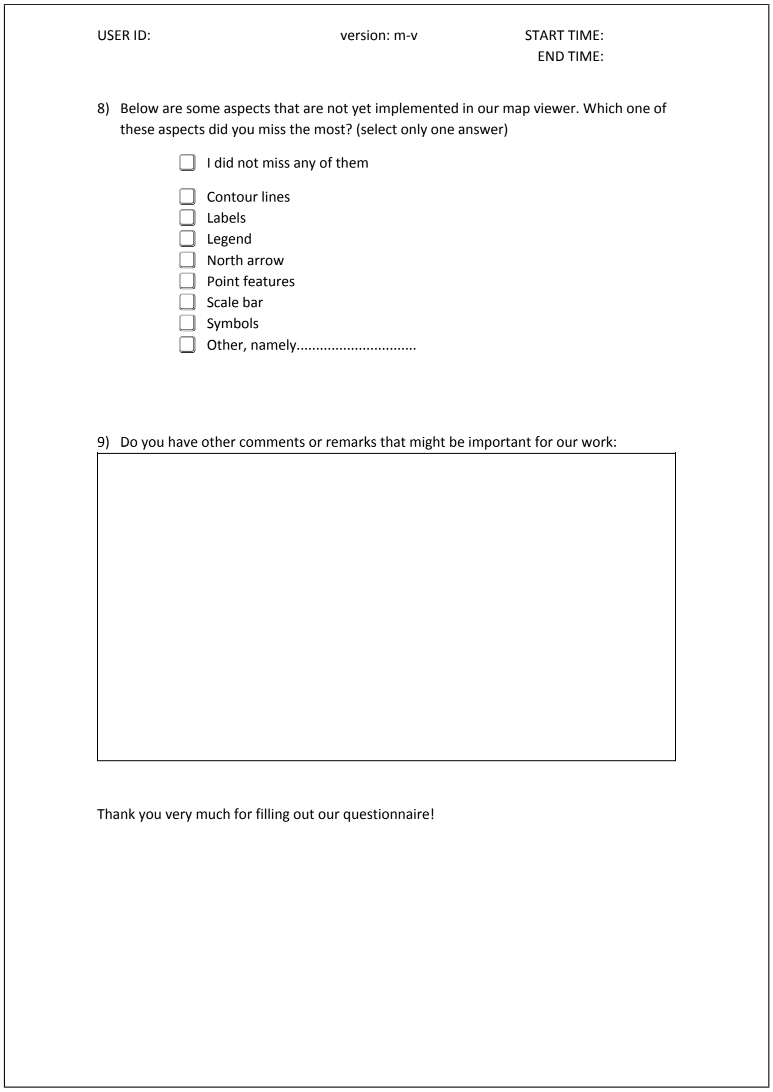
Strongly-Disagree	Disagree	Undecided	Agree	Strongly-Agree	I don't know
☐	☐	☐	☐	☐	☐

6) I did find the M map (the first) easy to work with.

Strongly-Disagree	Disagree	Undecided	Agree	Strongly-Agree	I don't know
☐	☐	☐	☐	☐	☐

7) I did find the V map (the second) easy to work with.

Strongly-Disagree	Disagree	Undecided	Agree	Strongly-Agree	I don't know
☐	☐	☐	☐	☐	☐



USER ID: version: m-v START TIME:
END TIME:

8) Below are some aspects that are not yet implemented in our map viewer. Which one of these aspects did you miss the most? (select only one answer)

☐ I did not miss any of them

☐ Contour lines
☐ Labels
☐ Legend
☐ North arrow
☐ Point features
☐ Scale bar
☐ Symbols
☐ Other, namely...............................

9) Do you have other comments or remarks that might be important for our work:

Thank you very much for filling out our questionnaire!

Responses in questionnaire

user id	dataset first	1) G	2) age	3) use	vario-scale * pleasant	vario-scale * easy	mutli-scale * pleasant	mutli-scale * easy	8) missing**	9) comments, remarks
the first round with smaller practice dataset										
1	multi	F	29	weekly	0	1	0	0	north arrow	I forget to remember the starting position
2	vario	M	50	daily	0	1	0	1	scale bar	I've missed map grid
3	vario	M	40	daily	1	1	1	1	symbols	I can't say the difference between maps
4	multi	M	30	daily	1	1	1	1	labels	annoying slow zoom
5***	vario	M	59	daily	0	-1	1	1	scale bar	
6***	multi	M	36	daily	1	1	-1	-1	labels	arrow key are inverted
7	multi	M	32	daily	1	1	2	2	labels	
8	multi	F	32	daily	0	-1	0	-1	0	I'd do the same testing after two weeks.
9	vario	M	29	never	-1	-1	1	-1	labels	
10	vario	M	30	occas.	0	0	0	0	scale bar	
11	vario	M	25	weekly	0	1	-1	-1	labels	
12	multi	M	39	weekly	1	1	1	1	labels	
the second round with bigger practice dataset										
13	vario	M	34	weekly	1	1	-1	-1	labels	
14	multi	M	29	daily	1	1	0	1	symbols	
15	multi	M	28	weekly	1	1	1	1	labels	
16	vario	F	31	daily	1	1	2	2	symbols	
17	multi	F	27	weekly	0	1	1	2	0	
18	vario	M	32	weekly	1	1	1	1	symbols	
19	vario	M	27	daily	1	1	-1	-1	scale bar	more continuous zooming, panning and features changes
20	multi	F	25	occas.	1	1	0	-1	symbols	
21	multi	M	28	daily	1	1	0	1	0	changing colour of area features
22	vario	M	35	weekly	1	1	1	1	scale bar	No difference between datasets
23	multi	M	32	daily	1	2	1	2	contour lines	
24	vario	F	31	weekly	1	1	1	1	labels	
25	vario	F	62	daily	2	1	1	-1	0	changing context after zooming out
26	vario	F	39	weekly	0	0	2	2	scale bar	

* strongly-disagree = -2, disagree = -1, undecided = 0, agree = 1, strongly-agree = 2
** I did not miss any of them = 0
***not recorded

Summary

Nowadays, there are many geo-information data sources available such as maps on the Internet, in-car navigation devices and mobile apps. All datasets used in these applications are the same in principle, and face the same issues, namely:

I. Maps of different scales are stored separately. With many separate fixed levels, a lot of information is the same, but still needs to be included, which leads to duplication.
II. With many redundant data throughout the scales, features are represented again and again, which may lead to inconsistency.
III. Currently available maps contain significantly more levels of detail (twenty map scales on average) than in the past. These levels must be created, but the optimal strategy to do so is not known.
IV. For every user's data request, a significant part of the data remains the same, but still needs to be included. This leads to more data transfer, and slower response.
V. The interactive Internet environment is not used to its full potential for user navigation. It is common to observe lagging, popping features or flickering of a newly retrieved map scale feature while using the map.

This research develops principles of variable scale (vario-scale) maps to address these issues. The vario-scale approach is an alternative for obtaining and maintaining geographical data sets at different map scales. It is based on the specific topological structure called tGAP (topological Generalized Area Partitioning) which addresses the main open issues of current solutions for managing spatial data sets of different scales such as: redundancy data, inconsistency of map scales and dynamic transfer. The objective of this thesis is to design, to develop and to extend the variable-scale data structures and it is expressed as the following research question:

How to design and develop a system for vario-scale maps?

To address the above research question, this research has been conducted using the following outline: 1) Investigate state-of-the-art in map generalization. 2) Study development of vario-scale structure done so far. 3) Propose techniques for generating better vario-scale map content. 4) Implement strategies to process really massive datasets. 5) Research smooth representation of map features and their impact on user interaction. Results of our research led to new functionality, were addressed in prototype developments and were tested against real world data sets. Throughout this research we have made following main contributions to the design and development of a system of vario-scale maps. We have:

- studied vario-scale development in the past and we have identified the most urgent needs of the research.

- designed the concept of *granularity* and presented our strategy where changes in map content should be as small and as gradual as possible (*e. g.* use groups, maintain road network, support line feature representation).
- introduced line features in the solution and presented a fully-automated generalization process that preserves a road network features throughout all scales.
- proposed an approach to create a vario-scale data structure of massive datasets.
- demonstrated a method to generate an explicit 3D representation from the structure which can provide smoother user experience.
- developed a software prototype where a 3D vario-scale dataset can be used to its full potential.
- conducted initial usability test.

All aspects together with already developed functionality provide a more complex and more unified solution for vario-scale mapping. Based on our research, design and development of a system for vario-scale maps should be clearer now. In addition, it is easier to identified necessary steps which need to be taken towards an optimal solution. Our recommendations for future work are:

- One of the contributions has been an integration of the road features in the structure and their automated generalization throughout the process. Integrating more map features besides roads deserve attention.
- We have investigated how to deal with massive datasets which do not fit in the main memory of the computer. Our experiences consisted of dataset of one province or state with records in order of millions. To verify our findings, it will be interesting to process even bigger dataset with records in order of billions (a whole continent).
- We have introduced representation where map content changes as gradually as possible. It is based on process where: 1) explicit 3D geometry from the structure is generated. 2) A slice of the geometry is calculated. 3) Final maps based on the slice is constructed. Investigation of how to integrate this in a server-client pipeline on the Internet is another point of further research.
- Our research focus has been mainly on one specific aspect of the concept at a time. Now all aspects may be brought together where integration, tuning and orchestration play an important role is another interesting research that desire attention.
- Carry out more user testing including; 1) maps of sufficient cartographic quality, 2) a large testing region, and 3) the finest version of visualization prototype.

Samenvatting

Tegenwoordig zijn er veel geo-informatie data bronnen beschikbaar, zoals kaarten op het internet, in auto navigatie systemen en in mobiele apps. Alle data bronnen, die gebruikt worden in deze applicaties, hebben in principe dezelfde opzet en hebben belangrijke nadelen:

I. Kaarten met een verschillende kaartschaal worden apart opgeslagen. Veel geo-informatie in de verschillende vaste kaartschalen is hetzelfde maar is toch voor elke kaartschaal apart opgeslagen. Het gevolg is veel dubbele opgeslagen informatie.
II. Door redundante data in de verschillende kaartschalen kan dit leiden tot inconsistenties.
III. De tegenwoordig beschikbare kaarten bevatten significant meer detailniveaus (in het Engels Levels of Detail genoemd, gemiddeld twintig kaartschalen) in vergelijking met oudere kaarten. Deze verschillende kaartschalen moeten worden gemaakt, hoewel er geen optimale methode bekend is om dit te doen.
IV. Voor elke vervolgdata vraag van een gebruiker na een zoom actie blijft een significant deel van de informatie hetzelfde, maar in gangbare webmap systemen worden alle gegevens opnieuw verstuurd. Hierdoor is er veel data overdracht en een langzame reactietijd.
V. Het volledige potentieel van de interactieve Internet omgeving (met bijvoorbeeld de beschikbaarheid van HTML5 en WebGL) wordt niet benut tijdens het gebruik. Het is gebruikelijk om achterblijvende, ineens verschijnende objecten en flikkering van nieuw ontvangen objecten waar te nemen tijdens het gebruik van de kaart.

Dit onderzoek ontwikkelt de principes van een vario-schaal structuur om deze problemen aan te pakken. De vario-schaal aanpak is een alternatief voor het verkrijgen en beheren van geografische data bronnen met verschillende kaartschalen. De aanpak is gebaseerd op de specifieke topologische structuur genaamd tGAP (topological Generalized Area Partitioning) die de hoofdproblemen van de bestaande oplossingen voor het beheren van de ruimtelijke datasets op de verschillende kaartschalen aanpakt, zoals dubbele data, inconsistenties tussen de verschillende kaartschalen en een progressieve gegevens overdracht. Het doel van deze thesis is om de vario-schaal structuur verder te ontwikkelen en uit te breiden aan de hand van de volgende onderzoeksvraag:

Hoe een systeem te ontwerpen en ontwikkelen voor vario-schaal kaarten?

Om bovenstaande onderzoeksvraag te beantwoorden is dit onderzoek uitgevoerd aan de hand van de volgende onderzoeksopzet: 1) Onderzoek naar de nieuwste kaart generalisatie technieken. 2) Bestuderen van de ontwikkelingen binnen de vario-schaal structuur tot nu toe. 3) Voorstellen van technieken om betere inhoud voor de vario-

schaal structuur te genereren. 4) Implementeren van strategieën om enorme datasets te verwerken. 5) Onderzoek naar de geleidelijke kaartweergave na zoom-acties en hun impact op de interactie met de gebruiker. Resultaten van het onderzoek hebben geleid tot nieuwe functionaliteiten, deze zijn gebruikt in een prototype en getest met datasets afkomstig uit de praktijk. Tijdens dit onderzoek hebben we de volgende bijdragen geleverd aan het ontwerp en de ontwikkeling van de vario-schaal structuur. We hebben:

- de vario-schaal structuur bestudeerd en hebben de meest urgente onderzoeksdoelen geïdentificeerd
- het concept van granulariteit ontworpen en een strategie gepresenteerd om veranderingen in de inhoud van de kaart zo klein en geleidelijk mogelijk te maken
- lijn objecten geïntroduceerd in de oplossing (die tot nu toe alleen vlak objecten ondersteunde) en een geautomatiseerd generalisatie proces ontwikkeld dat het wegennetwerk behoudt in alle mogelijke kaartschalen
- een aanpak voorgesteld om vario-schaal datastructuren te genereren voor enorme datasets.
- een methode gedemonstreerd om een expliciete 3D Space Scale Cube (SSC) weergave te genereren vanuit de tGAP structuur, die kan zorgen voor een betere ervaring tijdens het gebruik van vario-schaal kaarten
- een software prototype ontwikkeld dat in staat is een 3D SSC vario-schaal dataset in zijn volle potentiaal te gebruiken
- een eerste gebruikerstest uitgevoerd.

Alle aspecten samen met de alreeds ontwikkelde functionaliteit zorgen voor een meer complexe en complete oplossing voor vario-schaal kaarten: zowel productie vario-schaal inhoud als gebruik hiervan. Gebaseerd op ons onderzoek, is het ontwerp en ontwikkeling van een systeem voor vario-schaal kaarten nu duidelijker (en dichterbij operationele inzet). Daarnaast is het nu makkelijker om de nodige stappen te identificeren welke moeten worden genomen richting een optimale oplossing. Onze aanbevelingen voor toekomstig onderzoek zijn:

- Een van de bijdragen is de integratie van wegen in de structuur en hun geautomatiseerde generalisatie in het proces. Integratie van meer infra objecttypes naast wegen verdienen de aandacht (spoorwegen, waterwegen, nuts netwerken).
- We hebben onderzocht hoe om te gaan met enorme datasets die niet in het hoofdgeheugen van de computer passen. Onze ervaringen bestaan uit datasets die een provincie of een deelstaat met miljoenen objecten bevatten. Om onze bevindingen te verifiëren zal het interessant zijn om nog grotere datasets te verwerken met miljarden objecten (een heel continent).
- We hebben kaartweergaven geïntroduceerd waar de kaartinhoud zo geleidelijk mogelijk veranderd. Het is gebaseerd op het proces waar: 1) Expliciete 3D geometrie uit de structuur is gegenereerd; 2) Een doorsnede van de geometrie is berekend; 3) De uiteindelijke kaart gebaseerd is op de doorsnede die is gemaakt. Onderzoek hoe dit te integreren in een server-cliënt pijplijn op het Internet is een ander punt voor toekomstig onderzoek.

- Onze onderzoeksfocus heeft zich steeds een specifiek aspect van het vario-schaal concept per keer geweest. Nu kunnen alle aspecten ook samengebracht worden waarbij integratie, afstemming en het harmonieus samenbrengen een belangrijke rol spelen. Dit is een ander interessant onderzoek dat de aandacht verdiend.
- Meer gebruikerstesten uitvoeren met: 1) kaarten met genoeg cartografische kwaliteit, 2) een groot test gebied, en 3) de betere versie van het visualisatie prototype.

Curriculum Vitae

Radan Šuba was born in Pilsen, the Czech Republic, on June 27th, 1986. In 2012, he finished his studies at the University of West Bohemia, in Pilsen, and obtained his Master degree in Geomatics. During his studies, he gained practical experience when we worked as a land surveyor.

In 2012 Radan was appointed as a PhD candidate at the Delft University of Technology, the Netherlands, under the supervision of Prof. dr. ir. P. J. M. (Peter) van Oosterom and dr.ir. B.M. (Martijn) Meijers funded by the Dutch Technology Foundation STW (project number 11185). He researched vario-scale maps, alternative solution for obtaining and maintaining geographical data sets on different map scales. He focused on R&D to expand and to enrich current prototype for automated map generalization. At the end of his study, he received OGT Award for usage of Dutch dataset in research.

Currently, summer 2017, Radan is involved as a guest researcher in the project applying the vario-scale techniques to Shenzhen geographic data, produce thematic maps and develop advanced functionality. It is supported by Shenzhen Research Center of Digital City Engineering in China.