Usability test plan for truly vario-scale maps

Radan Šuba, Mattijs Driel, Martijn Meijers, Peter van Oosterom, Elmar Eisemann

May 30, 2016

Abstract

The development of vario-scale maps, which provide continuous map scales, gradually changing cartographic content and smooth user interaction, has reached a state where user testing is possible and viable. Recently we have developed a prototype where the full potential of truly vario-scale data can be used and tested with users. It is based on the Space Scale Cube (SSC) encoding of 2D vector data with continuous level of abstraction on the Z axis. Intersecting the SSC and rendering the result is done in real time making efficient use of the Graphics Processing Unit (GPU) and gives a smooth impression to the user. Using our prototype an initial usability test is designed to validate our hypothesis 'the vario-scale concept will provide better understanding and more intuitive perception of the map data, resulting in faster and better task execution'.

1 Introduction

Current dynamic maps on the internet which are based on multi-scale representations are not optimal for good user perception. Even a simple zoom-in operation requires from users navigation through scale ranges, represented by multiple discrete level of details (LoDs), with different content and (styling) representations leading to abrupt transitions. This can lead to disturbance or less effective map navigation for users, resulting in mistakes or at least delays during map reading. Therefore there have been developed additional solutions that work around these issues such as blending between layers, morphing (Danciger et al., 2009), snapping to predefined levels (van Kreveld, 2001) or introducing more intermediate layers between the LoDs which results in either reduced accuracy or flexibility (Dumont et al., 2015).

On the other hand there exists the truly vario-scale concept introduced by van Oosterom et al. (2014). It assumes that results of map generalization actions can be stored in specific data structure called tGAP (topological Generalized Area Partition) (van Oosterom, 2005; van Oosterom and Meijers, 2011). It makes the whole map repeatedly simpler and simpler where the least important feature in the map is simplified based on a global criterion. These incremental generalization steps are captured in the structure with minimal redundancy. Both the detailed objects at the largest scale and the intermediate objects generated during the generalization process are stored in a set of database tables. Next, the third dimension is used to encode the scale (or importance ranges of the tGAP representation), resulting in a volumetric partition of stacked polyhedra within the so-called SSC (Space Scale Cube). Maps are created by horizontal slices and continuous generalization corresponds to gradually moving the slicing plane (and scaling the clipping window). The tGAP/SSC structure can be used for generating all scales in a more smooth digital way then has been seen before.

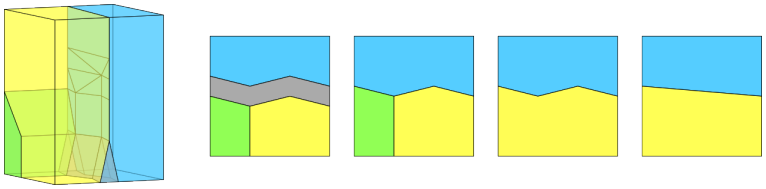


Figure 1: Example SSC data with horizontal intersections. In this example, the colors blue, grey, yellow and green respectively represent water, road, farmland and forest terrains.

The concept of vario-scale has been proposed and proven with various prototypes as described in multiple publications; see Section 2. These studies have provided theoretical and technical aspects of the solution in general. Section 3 describes our GPU-based implementation of the vario-scale prototype, called Intersector, based on slicing the SSC. However, a full understanding of how this vario-scale functionality affects user perception of a map is still missing. We believe that the vario-scale concept will provide better understanding and more intuitive perception of the map data leading to faster results and decisions of better quality. Since a usability testing is required to validate such an assumption, this paper presents the design of such a usability test; see Section 4. We want to find out whether vario-scale maps can indeed help users to maintain a better mental model of the world, compared to fixed level of detail maps. Finally, Section 5 mentions additional challenges which can be addressed in the near future.

2 Related work

Map generalization research of vario-scale has shifted towards a truly smooth vario-scale structure for geographic information. 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 stored as a single 3D (2D+scale) data structure: the SSC, see Figure 1.

Here, we further discuss the most relevant work. For an overview of smooth abstraction representations, we refer to (van Oosterom and Meijers, 2014). This paper also describes the Space Scale Cube in more detail, the first model to integrate vario-scale directly into its own structure. Although a general partition can be assumed (as represented by a 3D topological structure), in practice, the structure is usually represented by a set of polyhedra, which is also the assumption we will make. The original SSC proposal included an explicit boundary calculation to perform the slicing with a plane. The intersection 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 differing workloads, depending on the polyhedras' complexity.

To some degree, the approach resembles Constructive Solid Geometry (CSG) for manufactured products and virtual environments. With CSG, solids are defined as a series of boolean operations on geometric primitives, which results in the main cost when compared to mesh-based modeling. In 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 pixel-precise, which is sufficient. They do this by taking advantage of graphics hardware to do calculations at the level of pixels instead of primitives.

Taking perspective on usability testing some researchers faced very similar issues. There have been several debates between experts from different fields about how efficient smooth

change in content (animations in general) are. In that sense the most significant work is conduct by (Midtbø and Nordvik, 2007). They investigate how the dynamic, continuous, smooth zoom is perceived by users and compared these observations with a 'classic' map of more abrupt steps. They demonstrated clearly that the continuous zoom gives a better understanding for end users of locations depicted on a map during the zoom in and out operation. They approached vario-scale by many multi-scale maps together with flash animation, but still with discrete steps the opposite to our true continous vario-scale content.

More and more map use is on mobile devices. These devices have limited screen sizes and therefore even a more frequent need for good zooming (than desktop based map use). This was researched in (van Elzakker et al., 2008; Midtbø, 2010). In the study, Delikostidis et al. (2016) conducted a quantitative usability test in the streets of Amsterdam and pointed out that test subjects appreciated continuous map more than a static map. They claim that envisaging an overview map while looking at a detailed map view was easier than the opposite. However, frequent zooming was still needed to maintain the mental connection between the two.

In general, researches are more focused on testing new tools or map environments than testing the user perception of the different map content and interaction possibilities. However most of them agreed on the fact that the use of different kinds of animations, smooth zooming or other additional visual effects is growing in digital cartography and that people find such effects attractive.

3 GPU based vario-scale viewer

This section describes the concept of SSC intersection pixel rendering approach and how it is implemented on a GPU. A number of attractive and efficient to realize additions that can be used specifically with this approach are explained as well. 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. However, more detailed information about Intersection can be found in (Driel et al., 2016).

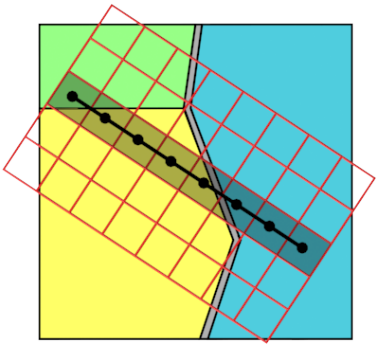
3.1 Concept

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 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 2.

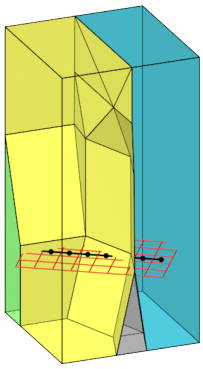
We first explain an alternative naive intersection method, as a way to explain by contrasting it with our own. The naive method will go over all pixels and for each pixel, do a point-in-polyhedron test for all polyhedra in the data. With the degenerate case of a pixel lying on a polyhedron boundary, 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.

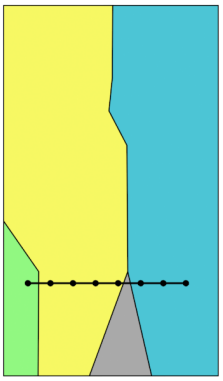
Building on this observation, we view the SSC orthogonally from the top downwards. We



(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 2: The intersection method's goal is to determine what 'color' to give each pixel in a raster. To do this, the center points of the pixels are tested for what color 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.

then render insides of the polyhedra below the raster, while simultaneously clipping off everything above the raster.

What we essentially do here is picking the z-axis as the projection axis, and by setting the view as orthogonal, all pixels are projected along the same axis. Then, rendering only below the raster will prevent any geometry above from interfering with the result below the raster. An example of this is shown in Figure 3.

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 will not happen with valid data.

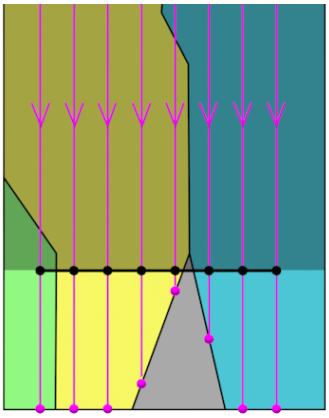


Figure 3: By projecting the points downwards we find the correct colors for the pixels. The greyed area must be explicitly ignored, otherwise some invalid colors will appear. In this example, if the greyed region was not clipped the first and fifth pixels would incorrectly get assigned yellow and blue respectively.

3.2 Implementation

The main argument of 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 othogonal downwards viewing resulting into 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 color mapping technique that maps IDs to colors, which gives us our final displayable image.

3.3 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 antilaliasing (for sharper images), non-horizontal intersections (for mixed scale representations), near-intersection blending (for dynamic movement zoom-feeling), and texture mapping (for enhanced map readability).

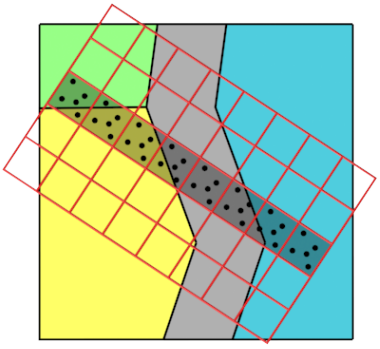


Figure 4: Using a form of SSAA, multiple samples are taken at every pixel, and their colors blended.

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 subpixel 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 4.

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 5 shows an example using a curve shaped surface, which results in high detail near center 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.

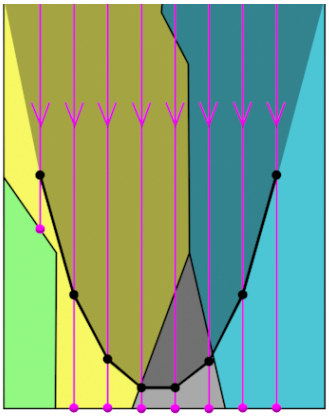


Figure 5: In this example, the center of the raster requires less abstraction than the raster edges. As a result, the forest becomes invisible and the road visible.

Near-intersection blending When quickly zooming through much information with our approach, some changes might still feel abrupt; e.g. one area removed and aggregared with neighbor. 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). 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 color) is defined. If we blend the colors of both the intersecting and near-intersecting terrains with a factor of $f_{blend} = \frac{d}{c}$, we smoothen 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 color 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 7. It does not influence anything other than the rendered color, terrain IDs are kept intact.

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

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 8. 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.

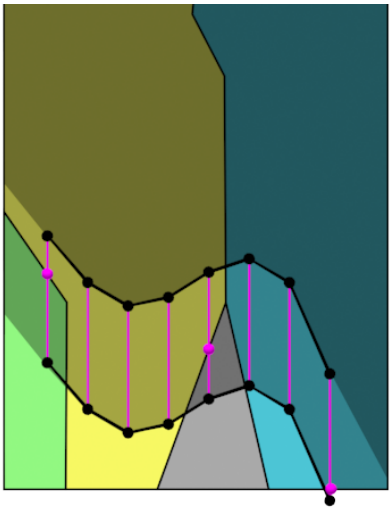


Figure 6: The three marked points represent an intersection within the given distance. The left and middle points will blend yellow-green and yellow-grey respectively. The right point intersects on the SSC boundary, so blending will not occur there.

3.4 Implementation details

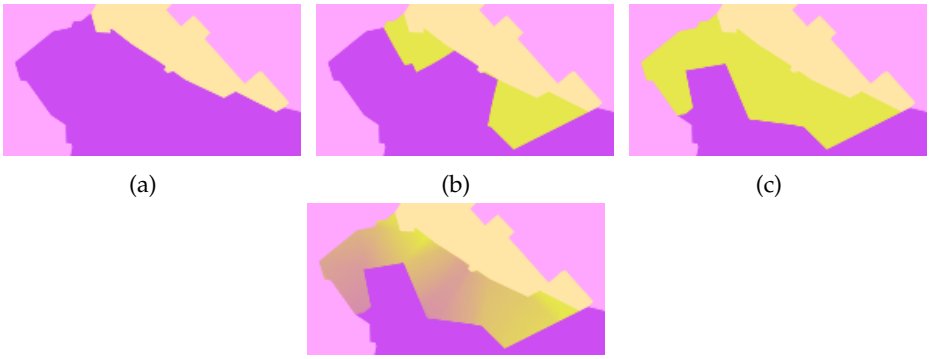
The Intersector prototype was written as an OpenGL ES 3.0 application in Java. Because of the use of the limited OpenGL ES feature set, the application should on mobile phone hardware with limited efforts.

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 10, along with some examples of them intersecting the rural region dataset. The control point surface is based on inverse distance weighting (Shepard, 1968), which is a method of determining a point's z value based on distance to, and the z-values specified by the control points. This method is an example of direct real-time manipulation of the intersection shape itself.

4 Design of experiment

With the Intersector prototype implemented the next logical step is to perform a usability test with users. Therefore, here we describe our initial plan for such a test. It will be carried out with a small group of users, where only the core of our vario-scale concept will be tested. On top of that, future additional testing can include more participants, additional functionality (as described in previous section: non-horizontal intersections, color blending, etc.) and different SSC content (range of generalization techniques and options possible to create the structure). Our usability test plan for spring 2016 includes the following elements:

Scope This usability test will indicate if vario-scale maps can provide better understanding and more intuitive perception of the data presented on the map to the users, compared to a fixed and limited number of maps for discrete map scales (the more traditional multi-scale approach). The main scope will be these two types of content. The Intersector is the tool which helps



(a) (b) (c)

(d)

Figure 7: This example shows how a transition from one color (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 noticable while in motion, scrolling through abstractions. Blending the colors (d) can make the transition seem less abrupt.

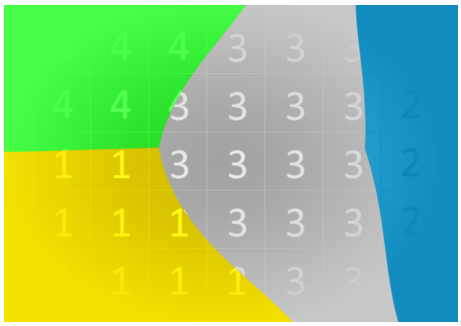


Figure 8: Texturing is shown as the numbers rendered in the regions. They fade on the sides because of the non-planar intersection used to generate the image, showing that abstraction variations influence the result.

us demonstrate our vario-scale approach. To eliminate the influence of the different working environment the experiment will be carried out with the same 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 (SSC based on tGAP content with smooth transitions), both covering the same geographic extent. We assume that users will 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?

Table 1: 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.

dataset	source	tGAP	Intersector	
	no. faces	no. faces	no. prisms	no. triangles
multi-scale	115.000	229.000	136.000	3.096.000
vario-scale			229.000	11.202.000

Apparatus All techniques described in the previous sections are implemented as a prototype and demonstrate their potential, but the implementation is not optimized to provide the most user friendly experience, which may negatively effect the results of experiment. For example, the user performs bad because he is irritated or confused by the set of mouse interactions and key strokes to use. Therefore we propose to use a simplified interaction with the Intersector prototype. We will reduce controls to four keys for movement and two keys for zoom in and out operation (no mouse). Specifically, arrow keys for movement and *i* and *o* keys for zooming in and out.

For executing the experiment one notebook, powerful enough to run the Intersector will be used by one test person (TP) in a session of maximum 15 minutes. The Intersector will run as a window fixed in size, only showing a map view (no additional controls on screen). The equipment will be set up in a quiet room to create a comfortable experiment environment where the TP will accomplish the task. The instructor will be present throughout the experiment for giving an introduction and in case any non-standard situations occurs.

The test methodology consisted of questionnaire, thinking aloud with audio-video observation and synchronized screen recording. This combination will allow further detailed investigation of TP's behavior, thinking and potential hurdles during the test session.

For this experiment, we have already generated two main datasets; First, discrete multi-scale, represents a classic approach, where 7 discrete scales are generated based on the well-know data factor 2 (in each dimension), i. e. level 0 corresponding 1:10.000 scale with 115.000 features, level 1 corresponding 1:20.000 scale with 28.750, level 2 corresponding 1:40.000 with 7.182, etc. Second, a vario-scale dataset, 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 20x20 km, see Table 1. Figure 9 shows a subset of the data. The structure has been generated with merge operation and simplification of object boundaries.

Procedure / tasks Before the experiment the TP will be welcomed and the experiment will be briefly introduced. Then the TP will spent few minutes by randomly using Intersector to 'get a grip' about the working environment. This will be done mainly to eliminate the factor of new unfamiliar environment. Several types of tasks based on map reading are specified: orientation, searching, analysis, routing, and planning tasks. When the experiment will start, the environment will be reset in default zoomed in location of discrete dataset. The TP tasks will then be:

1. Orientation-task: zoom out completely, while half-way also some panning in multiple direction, to see whole extent of the dataset and then try to get back as accurate as possible to the same location and zoom level as started.
   The time and the precision (the position of the screen) is captured.

2. 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.

3. 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.

4. Routing-task 1: find 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.

5. 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.

6. Analysis-task 2: Estimate distance between two churches that are somewhat further apart.
   The time and the closeness to the real distance is captured.

7. Planning-task: design a running track of about 10 km, which includes among others about 20% urban area and 30% forest track.
   The time and the correct quality (length and composition) of the track is assessed.

At the end of every task, the correct answer is given in form of coordinates and zoom level and the user should check if right location/scale is reached (and if not, then move this specified location). This task checking by user, is not part of the usability test, but just to make sure that users is at the right position to start next task. After the last task, the same series of tasks will then be repeated with the vario-scale dataset but with difference instances/cases. Half of the TPs starts with the multi-scale data and other half starts with vario-scale data,

Usability evaluation method The experiment will be captured in questionnaires and audio-video-recording of both the screen and the TP will take place. Furthermore, we will ask the TP to talk-out-loud.

The aim of the questionnaire is to get a more extended profile of the TP together with his personal opinion. The TP will be asked to fill the first part of the questionnaire before the test about the TP his/her background and GIS experience so far. At the end of the test session the TP will fill the rest of the questionnaire concerning the validation of the session. The second part of questionnaire where self-reported participant ratings for satisfaction, i.e. ease of use of the two types of data, will also be captured. The TP will rate the measure on a 5 point scale and it will be used for quantitative measures. Last part of the questionnaire where they can indicate likes, dislikes, suggestions and recommendations will provide subjective measures and their preferences. This data can be then used for planning additional usability testing in the near future.

Other quantitative metrics will be 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, for example, the difference in distance between real world position of starting and ending point for a given task.

On top of that, the recording should reveal TP's thinking and reasoning behind the TP's actions to give an indication where possible confusion is coming from. These aspect will detect qualitative aspects in our experiment.

5 Discussion and Future Work

This paper has presented our visualization prototype for vario-scale data which makes it possible to conduct an initial usability test. We proposed the layout of this test in the second part of the paper. The initial testing will take place during the spring 2016. We hope that vario-scale concept will prove to be more effective and faster in navigation for users. The first results of the usability testing are expected to be presented at the ICA workshop, June 2016, Helsinki. A positive result of testing will be a milestone confirming our main idea, however only starting point in the research of user perception of vario-scale, because there are still other options for even better smooth impression that will require more investigation. Therefore more testing should follow after our initial test and then should cover these issues:

- Different SSC cubes can be generated and tested. So far only one vario-scale SSC dataset have been introduced originated from one tGAP structure. Nevertheless either the creation of tGAP structure or the conversion from tGAP to SSC can be done in a multitude of ways. Different generalization techniques with different generalization operators is an example of the first. The conversion of all horizontal faces to the set of tilted triangles described by Šuba et al. (2014) is an example of the second.

- Alternative visualization technique can be used. Van Oosterom and Meijers indicate the possibility of more varied intersection shapes, resulting in mixed-scale representations, such as one that produces a fish-eye (local magnifier) or a perspective view. They are now possible in the prototype by choosing non-horizontal intersection planes, cf. Figure 10. Besides that there are other rendering techniques that were already implemented, namely near intersection blending and texture mapping should be tested.

- Different zooming techniques exist for even smoother impressions. The current implementation provides a map (slice of the cube) in any arbitrary location (scale) in real time. When a button is pressed, a new slice is determined. The key to provide smoother impression to user is the transition process between such as slices. User need some notion of acceleration, a change in speed is crucial, otherwise it feels stiff, mechanical and more abrupt (Penner, 2002). For example, zooming operation should may feel fast at the beginning, slow at the end or other way around. This and other transition techniques should also be tested.

- The main limitation of the current Intersector prototype is its current inability to stream data from a remote or local source. All data are loaded at program start instead of progressively, as would be expected in a server-client architecture. We want to realize a (web-based) server-client system with communication based on efficiently transferring the relevant SSC data (and using caching at a client).

- Larger datasets (than the current 20×20 km data) should be used. This is only possible when the server-client version of prototype has been realized (as larger data sets will not fit completely in memory of GPU). With larger data sets (complete province or country) also more fixed scales in the multi-scale approach are needed. And the user is expected to navigate over a wider range of scales (with even larger expected benefits for a vario-scale approach).

- Different types of data sets should be used in the multi-scale/ vario-scale usability testing (now the test is using topographic base data), but other types of data and other tasks should be defined; e.g. soil map or land cover map (Corine).

- Larger group of test users should be involved from various categories (including children).
- More advanced usability testing tools should be applied; e.g. using an eye-tracking system and analysing the eye movement (trajectories on screen), pause positions (possibly because of confusion at side of user), hot spot analysis (e.g. is used positioning the map in such a way that the focus is usually near the middle of the screen).
- Different and more interaction with extended functional-richer version of Intersector; e.g. use mouse to zoom and pan (mouse push-button, move, release-button), have identify and get feature attributes functionalities (either by clicking or by move over), have a routing computation in user interface, have a query composer to select specific features (based on attributes), etc.
- Instead of desktop based usability testing also mobile usability tests 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 natural manner.
- Also have text labels in the map.

In nearly all of these future activities the intention will be to compare the state of the art multi-scale based approach with the novel vario-scale approach.

6 Acknowledgement

This research is supported by the Dutch Technology Foundation STW (project number 11185), which is part of the Netherlands Organization for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs.

References

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

Delikostidis, I., van Elzakker, C. P., and Kraak, M.-J. (2016). Overcoming challenges in developing more usable pedestrian navigation systems. *Cartography and Geographic Information Science*, 43(3):189–207.

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*.

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.

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.

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.

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.

Midtbø, T. (2010). Consistency in maps with altering scales - a cartographic experiment by the use of mobile phones. *ISPRS Archives - Geospatial Data and Geovisualization: Environment, Security, and Society Special Joint Symposium of ISPRS Commission IV and AutoCarto 2010*, XXXVIII:1–6.

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.

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.

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.

van Elzakker, C. P. J. M., Delikostidis, I., and van Oosterom, P. J. M. (2008). Field-based usability evaluation methodology for mobile geo-applications. *The Cartographic Journal*, 45(2):139–149.

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

van Oosterom, P. (2005). Variable-scale Topological Data Structures Suitable for Progressive Data Transfer: The GAP-face Tree and GAP-edge Forest. *Cartography and Geographic Information Science*, 32(11):331–346.

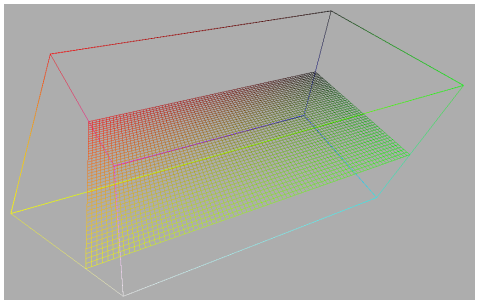
van Oosterom, P. and Meijers, M. (2011). Towards a true vario-scale structure supporting smooth-zoom. In *Proceedings of 14th ICA/ISPRS Workshop on Generalisation and Multiple Representation*, pages 1–19, Paris.

van Oosterom, P. and Meijers, M. (2014). Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data. *International Journal of Geographical Information Science*, 28(3):455–478.

van Oosterom, P., Meijers, M., Stoter, J., and Šuba, R. (2014). In Burghardt, D., Duchene, C., and Mackaness, W., editors, *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, number XV in Lecture Notes in Geoinformation and Cartography, chapter Data Structures for Continuous Generalisation: tGAP and SSC, pages 83–118. Springer.

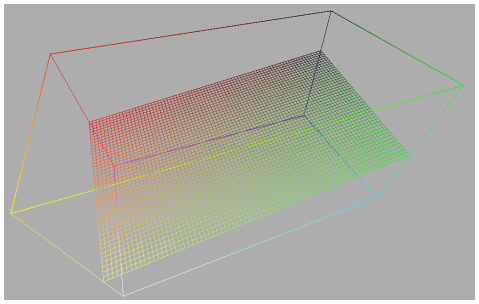
Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014). An area merge operation for smooth zooming. In Huerta, J., Schade, S., and Granell, C., editors, *Connecting a Digital Europe Through Location and Place*, Springer Lecture Notes in Geoinformation and Cartography, pages 275–293. Springer International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

Figure 9: Subset of the testing data (source: TOP10NL, near Maastricht, NL), shown at the lowest level of abstraction.



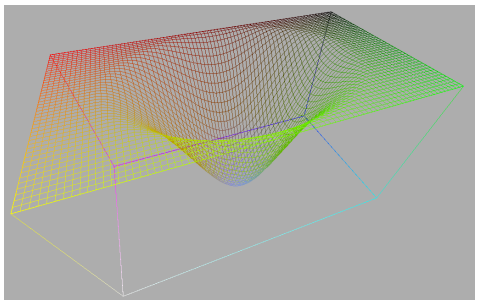
Horizontal raster.

Abstraction equal everywhere.



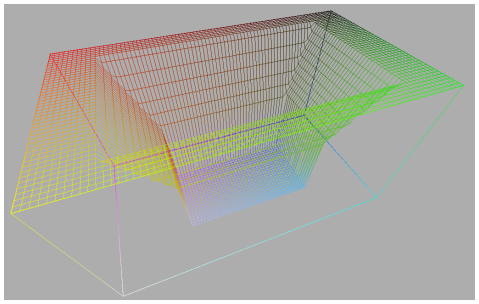
Nonhorizontal raster.

Abstraction lower towards the bottom left.



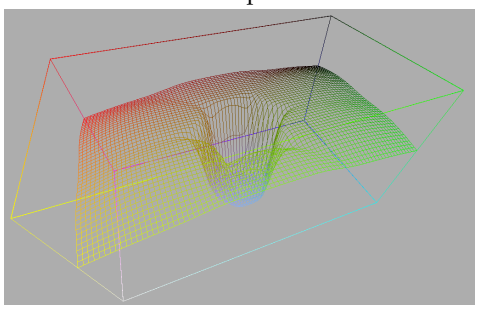
Sine curve raster.

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.

Figure 10: Rasters on the left produce corresponding intersections shown right. With lower abstraction, more buildings and roads become visible.