Using a generic spatial access method for caching and efficient retrieval of vario-scale data in a server-client architecture

Adrie Rovers
Delft University of Technology
Julianalaan 134
Delft, The Netherlands
a3rovers@gmail.com

Martijn Meijers
Delft University of Technology
Julianalaan 134
Delft, The Netherlands
b.m.meijers@tudelft.nl

Peter van Oosterom
Delft University of Technology
Julianalaan 134
Delft, The Netherlands
p.j.m.vanoosterom@tudelft.nl

Abstract

This paper presents a methodology for using a generic data-driven spatial access method as a communication mechanism for vario-scale data in a server-client setting. As a complete data set is often quite large, it is managed at the server side and supporting different scale levels is important. We show that a generic R-tree like grouping method, commonly used for efficiently organizing and retrieving data from a database, can be used in a networked architecture and that it allows off-loading processing tasks from a server to a client. This helps in making web services more scalable. The method supports efficient retrieval of partial data by a client and makes it possible to reuse data by means of caching. This can make communication more efficient.

Keywords: vario-scale data, vario-scale maps, spatial access methods, server-client architecture, caching

1 Introduction

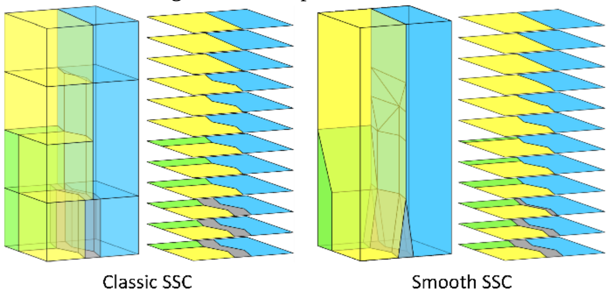
Geographic information is used to solve a diversity of problems in various application areas. Depending on the application, different spatial models are used to represent reality. They can include 2 or 3 spatial dimensions. For ages cartographers used 2D maps to model the shape of the earth, and more recently 3D models are being used for the analysis, simulation, and visualisation of our environment.

An important aspect concerning geographic information is the amount of detail that is captured in the model. Is a road represented as a line or as a polygon? Do we simplify certain features or are they not relevant for the application domain and not modelled at all? These considerations are commonly captured in discrete levels of detail (LOD); for each fixed scale a separate layer of geographic information is stored (Meijers, 2011). However, some data is stored redundantly as objects might exist at multiple scale levels. In addition, consistency is difficult to maintain because changes on one scale level should propagate to the next.

1.1 Vario-scale data

Instead of storing separate layers for each discrete scale, a spatial model could also describe a continuous LOD. Such a model is described in van Oosterom and Meijers (2013) and van Oosterom et al. (2014), where scale is represented as a 3rd dimension. A 2D base map is generalised and the results are stored in a single 3D structure, called the Space Scale Cube (SSC). The objects in this model have an importance range. This range describes their suitability for a certain LOD (classic SSC). Alternatively, they can be represented as polyhedra that gradually fade or aggregate in the 3rd dimension (smooth SSC). See figure 1.

Figure 1: The Space Scale Cube



Source: Adapted from van Oosterom et al. (2014).

1.2 Requesting a map over a network

To disseminate geo-information a distributed system can be used. Huang et al. (2016) showed that vario-scale structures can be used in a server-client architecture and that it is possible to request maps at arbitrary scale. However, transferring data takes time, which affects the responsiveness of the system, and sometimes costs can be involved for every byte that is send over the network. It is apparent that redundant data transfers, that is sending the same data multiple times over the network, should be avoided as much as possible.

In the scenario where separate geographic datasets are maintained for each discrete scale level these redundant data transfers are unavoidable as some objects might exist at multiple scales. Requesting more detail leads to the retrieval of a completely new dataset for a selected geographic region.

Having vario-scale data structure, the opportunity arises to reuse data that is already present on the client. When requesting a new map, it should be possible to reuse previously retrieved data, only request missing data, and make at the client side a complete map by combining the new response with previously cached responses. A communication method is needed that uses the client cache and that supports retrieval of partial vario-scale data from the server, while keeping the service scalable and responsive.

This paper shows that a generic data-driven spatial access method can be used as a mechanism for (partial) retrieval of vario-scale data in a server-client architecture, as described in Rovers (2016). We show that the Hilbert R-tree, which is commonly used for organizing and retrieving data from a database, can be employed in a server-client setting and that it can help make web services more scalable.

Section 2 gives a short theoretical background by introducing spatial access methods and the Hilbert R-tree. Section 3 describes the methodology. Section 4 describes our proof of concept implementation and shows the results of a benchmark used to assess the new method. Finally, Section 5 concludes the paper.

2 Spatial access methods

To implement a vario-scale model, data should be structured in such a way that it can be physically embedded in computer memory, ultimately stored as bits. Storage structures, indexes and compression techniques are needed. The fundamental issue for storage is that computer memory addresses are only 1-dimensional. This means that with the storage of spatial data some kind of mapping is needed. Gaede and Günther (1998) explain that there does not exist a mapping from n-dimensional to 1-dimensional space such that all objects that are close in reality are also stored close in 1-dimensional space. An ordering can only be imposed on a single dimension. Therefore, n-dimensional, and thus spatial data, require specialised structures in order to be used efficiently.

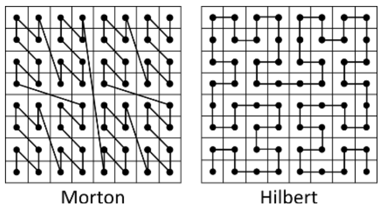
Methods that support efficient storage and retrieval of spatial data are commonly referred to as spatial access methods. A spatial access method applies both to spatial indexing as well as clustering (van Oosterom, 1999). An index helps in efficiently finding the right locations of data without having to perform a full search. It is a supplementary structure and therefore requires storage space in memory. Clustering has the goal to group data that is likely to be requested together on the same or nearby computer memory (disk pages) to minimize access time. This is a bottleneck in database performance. The minimal unit of transfer is often a disk page and without clustering a lot of transfers between memory and secondary storage might be needed.

2.1 Space Filling Curve

Clustering can be based on the organization of the index, but also space filling curves can be used for this purpose. A Space Filling Curve (SFC) can be used to group higher dimensional objects close together in 1-dimensional memory by imposing a linear ordering on the objects. This makes it possible to use common 1-dimensional indexing structures, such as the B-tree (van Oosterom, 1999).

Different SFC types exist. Figure 2 shows two common curves on a discrete 2D target domain. These are the Morton and the Hilbert curve. The SFC represents a path through a grid. The paths of SFCs are different and therefore some curves maintain better spatial proximity than others.

Figure 2: Two space filling curves

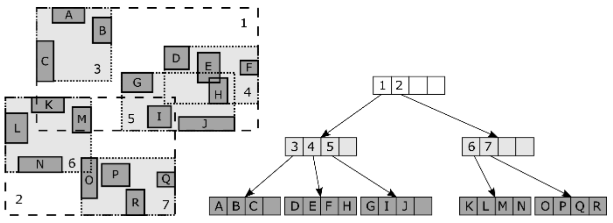


2.2 Bounding volume hierarchies

Bounding volume hierarchies build a tree structure on a set of objects. Pointers to the objects are typically stored in the leaf nodes of the tree. Higher-level nodes group lower-level nodes together and store a bounding volume that encloses the entire sub-tree. Bounding volumes of nodes may overlap. The tree is searched top-down by testing for overlap between the query geometry and the bounding volumes. If there is no overlap with a higher-level node there can also be no overlap with any of its children. The rest of the branch does not need to be searched.

The efficiency of the index depends on the algorithm that distributes the objects among the nodes. The common approach for creating these structures is by inserting the objects one by one in the tree (top-down). The objects are inserted in those nodes that need the least enlargement. The order of insertion has a large impact on the distribution. Well known examples are the R-tree (Guttman, 1984) and its variants: R+ tree (Sellis et al., 1987), R* tree (Beckmann et al., 1990). Figure 3 provides an example: a 2D R-tree.

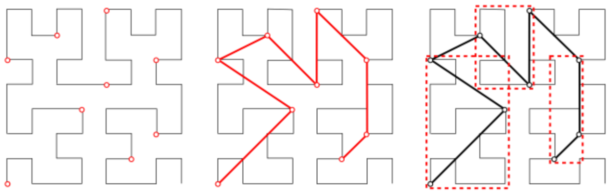
Figure 3: The R-tree



2.3 Hilbert R-tree

Another way to distribute the objects in a bounding volume hierarchy is by using a SFC. The Hilbert R-tree sorts objects, typically using the centroid, by their value on the Hilbert curve (Kamel and Faloutsos, 1994). Given this ordering, objects are grouped together into leaf nodes (Figure 4). The same goes for nodes. They are recursively grouped into higher-level nodes until the root-node is reached. The advantage of the Hilbert R-tree is that it is built bottom-up, giving a more compact tree.

Figure 4: Two-dimensional schematic example of sorting and grouping objects according to their centroid using a SFC.



3 Efficient partial data retrieval

The goal of this research is to achieve efficient communication, without too many redundant data transfers, for vario-scale data in a server-client architecture. It aims in minimizing network usage by grouping objects together and marking them explicitly as cacheable. It also aims in achieving scalability, i.e. the ability to facilitate many concurrent users, by providing the client the possibility to determine delta-requests, so that we can use the processing power of the client and off-load work from the server. A data-driven spatial access method is used to let the client retrieve partial data, based on the following objectives:

1. cluster data likely to be used together into packages on the server, based on *scale* and *geographic extent*,
2. let the client retrieve packages using a spatial index structure,
3. and use the client cache to re-use packages.

3.1 Requirements

To make communication suited for a server-client setting we place the following restrictions on the method:

- **Leaf nodes will refer to data packages:** This is usually the approach followed for databases, where the size of each leaf corresponds to the size of a disk page. The structure is used to make a sub-selection and to retrieve the disk pages with candidate answers from secondary storage. Precise computation on the objects takes place in memory. In a server-client setting similar conditions apply. The most time is spent on retrieving the data. Besides the actual time to transfer data there are per-interaction set-up costs for the TCP/IP stack and the headers in HTTP requests. Therefore, it is more efficient to transfer data in groups. Objects should be grouped in packages just as objects are clustered on a disk page.
- **Constant package size:** Packages should approximately have the same size when measured in bytes. A maximum threshold should be specified. This restriction is needed because the number of coordinates and therefore the size of geometry is variable. Just grouping an equal number of objects together does not give packages of constant size and would result in different transfer and processing times on the client. The responsiveness of the client would be unpredictable.
- **Full nodes:** Full nodes lead to a more compact index and is more efficient for transfer over a network.
- **Tree is balanced:** A balanced tree minimizes the worst-case search time of the index and makes the client more stable.
- **Axis aligned minimum bounding box:** Rectangles can be compactly encoded and allow fast filtering computations.

3.2 Clustering objects to create packages

An important consideration is what size the packages should be. On the one hand, large packages are better for reducing overhead costs. On the other hand, this means that additional candidate answers are retrieved that will not match the query. If a client must retrieve packages with a large spatial extent in relation to its viewport a lot of data may be transferred that is not directly needed. However, if a client stores the additional data in cache it is likely that it can be used for sequential queries.

To achieve efficient clustering the aim is to minimize the volume of and overlap between packages. They should be as compact as possible. This will increase the percentage of correct candidate answers that are retrieved over the network.

Clustering can be done using different techniques. However, in our proof of concept implementation we created the packages using a Hilbert SFC. For each object the value on the curve is calculated using its centroid (of 3D box: 2D spatial extents and 1D importance range). This value is used to impose a linear ordering on the objects. Groups are made based on this ordering.

3.3 Spatial index

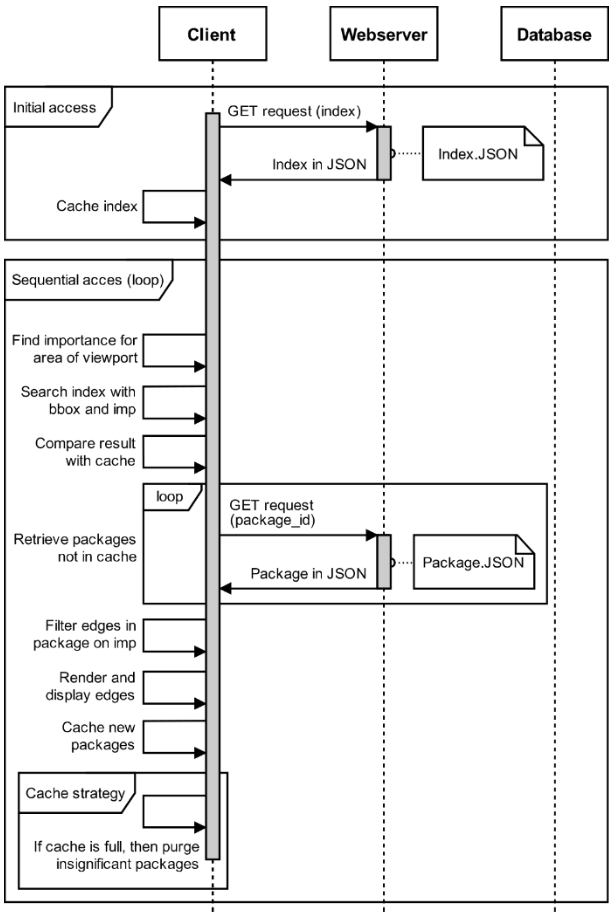
The Hilbert R-tree is used as a spatial index on the client. It is balanced and has full nodes. This makes the tree compact and efficient for searching. Furthermore, the tree is fast to build and easy to implement. Because a SFC is used to determine which nodes should be grouped together the method is generic and can be extended to higher dimensions.

3.4 Communication between client and server

As first step, the client retrieves the index and uses it to find the sub-selection of packages that contain candidate answers. The client retrieves the needed packages, either from cache or over the network, and processes them to find exactly matching objects. Figure 5 illustrates the communication steps. Initially, the client sends a single GET request for the index. Subsequently, it performs a series of actions for every new map it needs to construct. For every map query the client traverses the index using the bounding box of its viewport and its corresponding importance value (imp). The size of a viewport (in world coordinates) is an indication of the LOD that is needed, which can be translated to an importance value. All packages needed, as indicated by the index, are retrieved from the server, if they are not already in cache, and are filtered to get only the objects for the correct LOD. Finally, the objects are rendered and the newly retrieved packages are cached.

The packages and the index are identified as unique resources. They are placed on the file system, but can alternatively be stored in the database. A single GET request is made for every resource. This allows caching and the use of shared cache layers. Furthermore, it allows packages to be placed on different servers which makes it possible to add proxy servers or intermediary layers for load balancing.

Figure 5: Sequence diagram for the communication

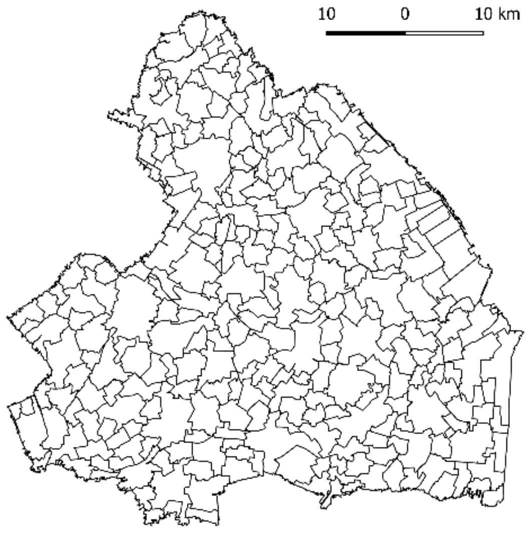


4 Proof of concept and Results

For the proof of concept a prototype client was developed that can communicate with the server using the new *Package-based* methodology. A classic SSC was generated from a topographic base map for the province of Drenthe in the Netherlands (Figure 6). The dataset has 1,110,123 edges and a total size of 625 MB. The edges were grouped into packages with a size of 500 KB. Details of the implementation are described in Rovers (2016). The code is available on github[1].

The new *Package-based* methodology for communication (referred to as option P) was assessed by comparing the prototype to the *Alternative* of retrieving ready-made maps (option A). The alternative is stateless, i.e. each request is made independent of any previous responses. The client simply requests a completely new map from the server for every interaction (panning, zooming, etc.). Reusing data that is already present on the client is thus not possible (similar to traditional multi-scale representations). Huang et al. (2016) describe this communication mechanism for retrieving ready-made maps.

Figure 6: A generalised map of Drenthe derived from the SSC



For the assessment we simulated different user scenarios (Rovers, 2016). The efficiency of communication varies. For some usage scenarios the amount of data transfers with a package-based communication was reduced, in other scenarios more data was needed. Figure 7 shows the sequence of queries for a typical scenario where using option P is beneficial.

We measured the data transfers (Figure 8) and the time till last byte (TTLB, cf. Figure 9) for both options. The total bytes sent over the network for option A is 2.5 MB, for option P this is 1.3 MB. The new method is thus more efficient regarding data transfers. This is also the case for TTLB, even for some queries where more data is transferred. There are two reasons for this: a. the packages can be requested in parallel, while we must wait until the entire map is constructed on the server for option A, and b. the server is less complex for option P and only has to send the requested packages, while for option A the server has to find which data to send (by means of a database query).

During interactive use the package-based methodology gets more efficient. With long sessions, where a user visits the same area multiple times, we can reduce the amount of data that is needed increasingly. If the user minimally pans the map, for option A completely new data is requested. For option P, it is likely that the map can be reconstructed using the packages that are already in cache (Figure 10).

[1] https://github.com/a3rovers/thesis/

Figure 7: Sequence of queries for the user scenario: zoom-in (1-14,17-19,21,31-33), pan (15,16,20,34), zoom-out (22-30).

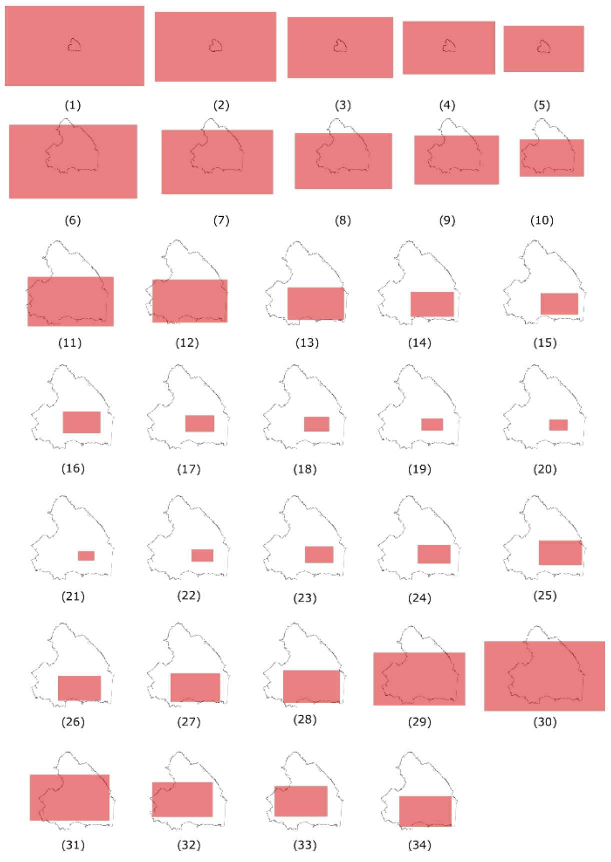


Figure 8: Data transfers. Note that with Option P data can be reused for subsequent requests.

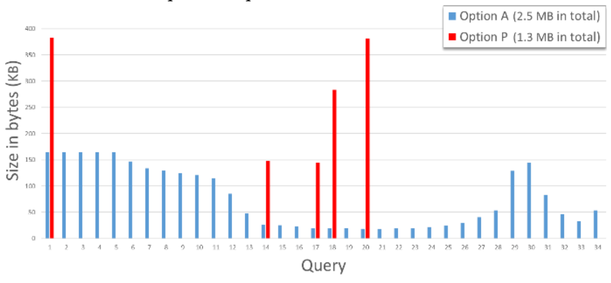


Figure 9: Response times

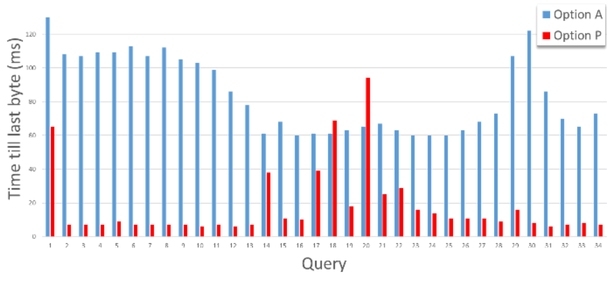
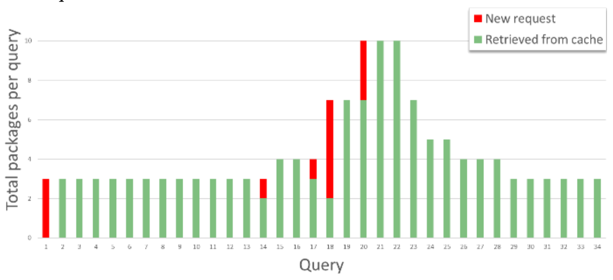


Figure 10: Total packages needed to make a map. Colours indicate whether a package is requested from the server, or can be requested from the local cache.



It should be clear that the performance of the method depends on the effectiveness of the algorithm that clusters the data into packages. If clustering is improved, also the efficiency of the new method is improved.

5 Conclusion

We presented a method for partial retrieval from a larger varioscale data set in a server-client setting based on a data-driven spatial access method. The method supports efficient retrieval of partial data by a client and makes it possible to reuse the data by means of caching. It also allows for a relatively simple server implementation, thereby off-loading work from a server to a client. This can help in making web services more scalable.

The Hilbert SFC was used for clustering and the Hilbert R-tree was used as an index on the client. It should be investigated if clustering can be improved or if different spatial access methods can be used. If clustering is improved, also the efficiency of the method will improve.

Also, clustering of the packages is dependent on the vario-scale source data. It is assumed the data is effectively generalised. In our test data, the geographic features had similar extents. However, large geographic features could affect clustering. In this case, it should be investigated if it is needed to cut up the geometry so that they can be distributed among different packages. Alternatively, the extents of the features could be used as an additional dimension in the calculation of the Hilbert key. This makes it more likely that large features are grouped together in the same package.

Furthermore, the method is generic for the way in which data is retrieved by a client. This gives support for the hypothesis that the method can also facilitate communication for the smooth SSC and other use cases with higher dimensional data, such as 4D point clouds (van Oosterom et al, 2015). It is expected that communication can be similar as with varioscale data, and that only the implementation of the filter and visualization steps need to be different.

References

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. (1990) The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM*

SIGMOD international conference on Management of data - SIGMOD 90. Association for Computing Machinery (ACM).

Gaede, V. and Günther, O. (1998). Multidimensional access methods. *CSUR*, 30(2):170-231.

Guttman, A. (1984). R-trees. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD 84*. Association for Computing Machinery (ACM).

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.

Kamel, I. and Faloutsos, C. (1994). Hilbert R-tree: An improved R-tree using fractals. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 500-509, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Meijers, M. (2011). *Variable-scale Geo-information*. PhD thesis, Delft University of Technology.

Rovers, A. (2016). Exploring the use of a generic spatial access method for caching and efficient retrieval of vario-scale data in a client-server architecture. Master's thesis, Delft University of Technology.

Sellis, T., Roussopoulos, N., and Faloutsos, C. (1987). The R+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, pages 507-518.

van Oosterom, P. (1999). Spatial access methods. In Goodchild, M. F., Longley, P. A., Maguire, D. J., and Rhind, D. W., editors, *Geographical Information Systems Principles, Technical Issues, Management Issues, and Applications*, volume 1. John Wiley & Sons.

van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., Tijssen, T., Kodde, M., and Gonçalves, R. (2015). Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. *Computers & Graphics*, 49:92-125.

van Oosterom, P. and Meijers, M. (2013). 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). Data structures for continuous generalisation: tGAP and SSC. In *Lecture Notes in Geoinformation and Cartography*, pages 83–117. Springer Science Business Media.