INSPIRE Multiple-Representation and Data Consistency Workshop, Ispra (VA, Italy) on 7-8 November 2006.

The tGAP structure: minimizing redundancy and maximizing consistency and offering access at any LoD

Peter van Oosterom

Delft University of Technology, OTB, Section GIS-technology,
Jaffalaan 9, 2628 BX Delft, The Netherlands.
Phone: +31 15 2786950, Fax +31 15 2782745.
Email: P.J.M.vanOosterom@tudelft.nl

1. Introduction

Despite the growing popularity of geo-information (maps) in all kinds of web services, such as routing, or providing (spatial/context) information for a given object, and also the growing use of mobile applications, there are currently severe problems with respect to the use of mobile maps. One of the main problems is due to the small screens of the mobile devices, not being able to show a lot of map. A user needs to pan and zoom a lot to get a sufficient spatial understanding, that is, a feeling of the sizes, directions and distances between the relevant objects and their context. With current technology in most mobile GISs (or Location Based Services, LBS) it is possible to zoom and pan. However, after a zoom or pan action, nearly in all cases a complete redraw is performed. The user now often looses the 'mental' contact between the two maps. Initial experiences show that the users are getting lost (and do not build a good mental map in order to support their task) and therefore do not appreciate the new solutions based on mobile maps. One of the key solutions to the described problem of user disorientation is vario-scale maps. There are a least two possible ways to use vario-scale maps:

- the first approach presents a map where the feature of interest (location or route) is shown at a higher detail level (large scale) and the surrounding features are shown with less detail (small scale), so this is a non-uniform scale representation within one image;
- the second approach is based on smooth zooming and panning: at a given moment the map has a uniform scale, but in an animation style of visualization, the map is continuously transformed into a target scale or location.

Both approaches try to help the user creating better mental maps, so they do not get disoriented. This is valid for either static (provide information for a certain object/location and its environment) and/or dynamic (provide information for a certain route between two locations and the environment) mobile GIS applications. Two use cases, a static and a dynamic one, are selected in order to investigate the effectiveness of the proposed solution with user tests. Before these tests can be executed, a prototype system has to be developed first. One of the important components of the prototype is the vario-scale data server being able to provide the necessary geo-information to the rendering engine. As the context is a mobile application, care has to be taken to use limited bandwidth. Therefore, the vario-scale data server should provide progressive transfer. That is, when zooming-in, only the refinements (additional detail for a larger scale representation) should be transferred to the mobile client and not a complete new map.

2. The tGAP structure and progressive transfer

Data structures supporting variable scale data sets are still very rare. There are a number of data structures available for multi-scale databases based on multiple representations (MRDB's), that is, data to be used for a fixed number of scale (or resolution) intervals. These multiple representation data structures try to explicitly relate the corresponding objects at the different scale levels, in order to offer consistency during the use of the data. Drawbacks of the multiple representation 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 scales. Another drawback of the multiple representation data structures is that they are not suitable for progressive data transfer as each scale interval requires its own (independent) graphic representation to be transferred. Nice examples of progressive data transfer are raster images, which are first presented relatively fast in a coarse manner and then refined when the user waits a little longer. These raster structures could be based on simple (raster data pyramid) or more advanced (wavelet compression) principles. For example, JPEG2000 (wavelet based) allows both compression and progressive data transfer from the server to the end-user. Also, some of the propriety formats such as ECW from ER Mapper and MrSID from LizardTech are very efficient raster compression formats based on wavelets and offering multi-resolution suitable for progressive data transfer. Similar effects are more difficult to obtain with vector data and require more advanced data structures.

The tGAP structure is a data structure supporting vario-scale vector data. In earlier research both the theoretical and practical (implementation) aspects of the tGAP structure (topological Generalized Area Partitioning) have been described (van Oosterom, 2005, van Oosterom, de Vries, Meijers, 2006). Purpose of this tGAP structure is to store the data only once, with no redundancy of the geometry, and derive different representations of this same data on the fly according to the level of detail needed. It has further to be explored how this vario-scale data server can be put to effective use in a mobile service/client environment.

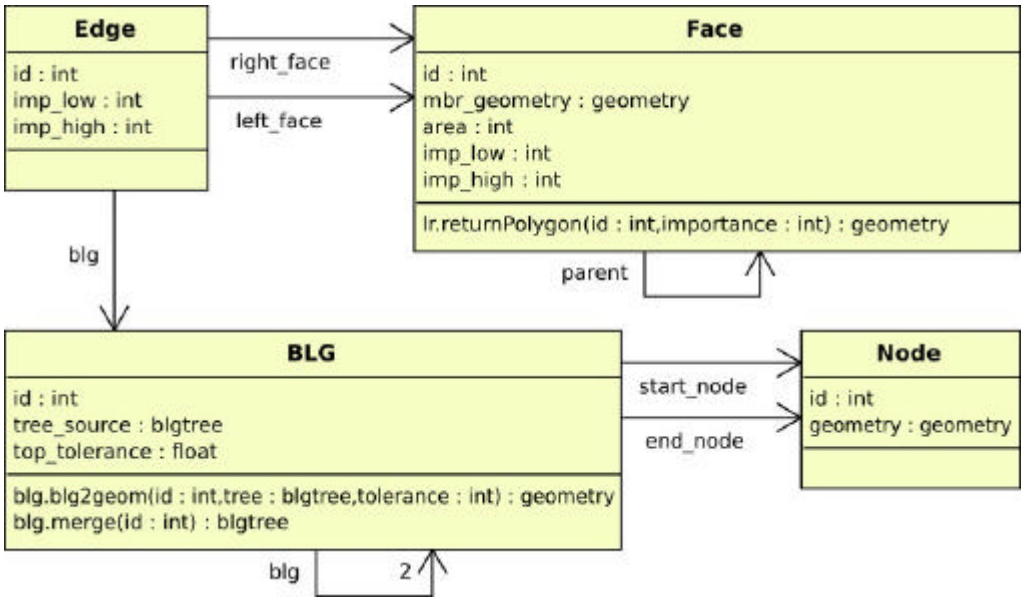

Figure 1: UML class diagram of tGAP tables, from (Meijers, 2006).

The tGAP structure consists of a few structures: the face tree that captures the merging of faces as the result of generalization via the parent-child relation of nodes in the tree; the edge forest that holds the boundary edges needed for any level of generalization, derived from their relation to the faces; the Binary Line Generalization (BLG) trees that enable the simplification of edges; there is a BLG tree for each edge. The tGAP structure is implemented in Oracle tables shown in Figure 1 and Appendix A.1 does give the corresponding SQL table definitions. A 3D spatial index (R-tree) is used for fast selection of features to be displayed for a given detail level: two dimensions are used for spatial selection (based on the bounding box) and the third dimension is used for specifying the required importance level; see Figure 2.

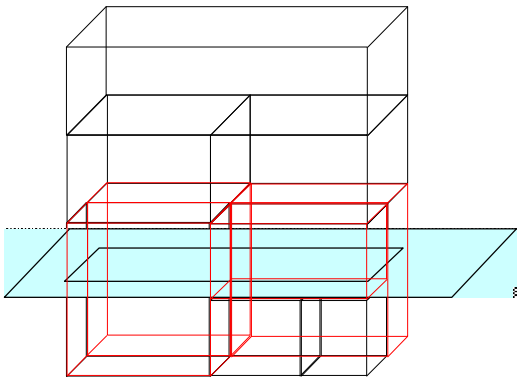

Figure 2: Importance levels schematically represented by the third dimension (at the most detailed level, bottom, there are several objects while at the most coarse level, top, there is only one object); from (Vermeij et al, 2003). The hatched plane represents a requested level of detail and the intersection with the symbolic '3D volumes' then gives the faces.

3. Current research

The tGAP structure can be used in a number of different ways: 1. to produce a representation at an arbitrary scale (a single map), 2. to produce a representation with feature(s) of interest at a larger scale and the surrounding features at smaller scales (a non-uniform scaled map) or 3. to produce a continuous range from rough to detailed representations. These three ways of using the tGAP structure are all useful, however the smooth zooming, realised through progressive transfer does seam to be the most promising solution for mobile maps. Currently research is being conducted to improve the tGAP structure with respect to the theoretical, functional, and practical aspects.

Improvement in the theoretical aspects include:

- More formal description of the structure (based on axioms), and how the different parts (tGAP face tree, tGAP edge forest, BLG edge trees) work together.
- Consider collapsing of areas in lines or points. As the current tGAP structure is designed for area objects only, this implies a (partial) reconsideration of the basic principles. For example, for a certain detail level a road area must collapse to its centreline, while its neighbour faces must then be extended until the road centreline (this situation was not yet possible in the current tGAP structure). An area may also collapse to a point, e.g. the centroid location, for a certain detail level. Again, the neighbour areas must then be expanded to cover for the collapsed area.
- Change the original principle of building the tGAP structured from a global minimum to possibly a local minimum, in order to localise the effects of an update to the whole structure. The original principle of creating the tGAP structure is to join the least important face with the most compatible neighbour. Two functions are used for calculating the face importance and compatibility between two faces. The change of values or formulas of these functions results in a different type of generalization (with a different filling of the tGAP structure). Important in the context of updates is the fact that a change of a face geometry may propagate through the whole structure. This is caused by the global minimum requirement. Therefore, we will investigate alternative tGAP structure creation/update principles.

Current research on functional aspects is focused on:

- Support the updating of the source data, without rebuilding the complete tGAP structure. We want to keep the changes as local as possible by controlling the propagation of changes in the structures. A list of possible updates is reshaping a boundary edge, splitting a face, merging two faces. These updates will affect the three parts of the tGAP structure, the face tree, the edge forest, and BLG trees. The change on the area size of faces will change their importance values, which in turn will affect the face tree. We expect the changes to be limited to a branch

of the tree, leaving unaffected the other part of the tree. The edge forest has a strong relationship with the face tree. By knowing which is the branch of the face tree that is affected by changes, we may determine the part of the edge forest that will be possibly affected by changes. The change in the BLG trees is limited to the edges affected by changes. There is no propagation affect here, because each edge has its own BLG tree.

- Support the queries for the different types of vario-scale representations (non-homogenous scale, feature centred map, vario-scale animation: smooth zoom/pan).

structure	*#face/Mb*	*#edge/Mb*	*#blg/Mb*	*#node/Mb*	*Total Mb*
Basic topology	170.368/2	418.530/94	-/0	281.216/11	107
tGAP structure	340.735/56	7.113.680/291	658.219/133	281.216/11	491

Table 1. Number of rows and size in Mb in tGAP structure for a sample data set (compared to the same data at the largest scale in a 'mean-and-lean' topology structure)

The current implementation is based on (Meijers, 2006), which provides a proof of concept of the tGAP structure. However, also a number of potential improvements were indicated. The storage space required by the current implementation is a bit worrying: about 5 times more space than a 'mean-and-lean' topology structure (according to the SQL table definition in appendix A.2) at the largest scale (fixed); see table 1. The use of more disk space itself is not dramatic, but the fact the accessing and transferring this information may also take more time is the main reason to investigate improvements in the data storage (and transfer) efficiency. Therefore, current work on practical aspects include:

- Reduce the storage requirements of the current implementation. Most of the overhead is for storing edge information. The edge table keeps information about the left and right face for each edge, as well as the edge importance values. An edge is in the boundary of at most two faces in the lowest level of detail. During the merging of faces for the creation of the face tree, the same edge will be a boundary edge of several faces in different steps of the process. A separate row is created currently for each version of an edge in a given step of the process. The different rows storing versions of the same edge have the same values for most of the fields. The current solution may be replaced with having only one row for each edge, and keep the information for each version of the edge in the corresponding fields as arrays of values, e.g. imp_low is an array of importance low values. The proposed new table definitions can be found in appendix A.3.
- Improve the time performance of the current implementation. The code for the first implementation was written in shell scripts calling PL/SQL procedures inside the Oracle database. This will be replaced with C or C++ code (that will execute faster as they are compiled languages), with possibly keeping some of the functions/procedures written in PL/SQL. However, the main performance gain is expected to be obtained by minimizing the number of SQL selections. Instead of many 'small SQL queries' the goal will be to have just a few 'large SQL queries' efficiently retrieving all necessary data. In theory these could be four queries, one for each of the based tables: face, edge, blg and node. The WFS server on top of the database then streams the result of the queries to the WFS client. This client then builds a local copy of the tGAP structure. After having received a few records the first rough visualizations can be made. At the same time this local structure is refined with additional details being streamed-in (enabling subsequent visualizations to be more refined).
- Modification of the standard communication protocols WFS/GML (Web Feature Server/Geography Markup Language) to enable progressive transfer of vario-scale data. In order to accomplish the progressive refinement when the geo-data is retrieved from the WFS-R service (our proposed special variant of the standard WFS service to provide progressive refinement) an 'order by' expression (or functional equivalent) has to be included in the WFS query to the data source. Necessary requirement for the data storage system is therefore that the WFS service can retrieve the geo-objects from the data source in a sorted way. A

'Progressive Refinement' WFS-R service, that serves data in order of importance and in a certain importance range, does not need large extensions to WFS software. What is extra in the WFS layer is: 1. adding a 'order by' to the queries sent to the data source and 2. adding a importance selection to the Filter conditions (either spatial or non-spatial) that the user already has specified in the request to the WFS. A tricky aspect is the fact that the ordered information is mix of the four involved tables (face, edge, blg and node).

Crucial for the quality of the tGAP-structure is the assignment of proper importance values to the involved feature classes (and importance function) and the compatibility values between two different feature classes (and compatibility function). More research is needed in this area to automatically obtain good generalization results for real world data. Related to this aspect is finding the relationship between the importance value and the actual scale.

4. Conclusion

In order to evaluate the effectiveness of the smooth zooming and the new vario-scale mobile maps, two use cases are selected. The current prototype will be enhanced to support these use cases and actual tests with end-users will be executed (see project plan on website for more details http://www.gdmc.nl/uwsm2). The enhanced prototype will first be based on a simulation of a mobile device: a small window in desktop environment. The user experiences will then be included in a second prototype, which will be realized on top of a true mobile platform. Again, users are asked to perform their tasks with the aid of this prototype and the effectiveness will be compared to the currently available mobile solutions (without vario-scale maps).

References

Meijers, B.M. (2006), *Implementation and testing of variable scale topological data structures - Experiences with the GAP-face tree and GAP-edge forest.* TU Delft, MSc Geomatics thesis, June 2006.

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 (4): 331-346.

van Oosterom, P., de Vries, M. and Meijers, M. (2006), Vario-scale data server in a web service context. In the Workshop of the ICA Commission on Map Generalisation and Multiple Representation – June 25th 2006.

Vermeij, M., van Oosterom, P., Quak, W., and Tijssen, T. (2003), Storing and using scale-less topological data efficiently in a client-server DBMS environment, In the *proceedings of the 7th International Conference on GeoComputation*, University of Southampton, Southampton, UK 8-10 September 2003.

Appendix A: Table definitions

A.1 tGAP structure

```
sql> desc tgap_face;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 face_id                                            number
 mbr_geometry                                       mdsys.sdo_geometry
 area                                               number
 imp_low                                            number
 imp_high                                           number
 parent_id                                          number

sql>  desc tgap_edge;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 edge_id                                            number
 left_face_id                                       number
 right_face_id                                      number
 imp_low                                            number
 imp_high                                           number
 blg_id                                             number

sql> desc tgap_blg;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 blg_id                                    not null number(11)
 start_node_id                                      number(11)
 end_node_id                                        number(11)
 child1_id                                          number(11)
 child2_id                                          number(11)
 tree_source                                        blgtree
 top_tolerance                                      float(126)

sql> desc tgap_node
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 node_id                                            number
 geometry                                           mdsys.sdo_geometry
```

A.2 topology structure

```
sql> desc face;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 face_id                                   not null number

sql> desc edge;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 edge_id                                   not null number
 left_face_id                                       number
 right_face_id                                      number
 start_node_id                                      number
 end_node_id                                        number
 geometry                                           mdsys.sdo_geometry

sql> desc node;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 node_id                                   not null number
 geometry                                           mdsys.sdo_geometry
```

A.3 Improved tGAP structure proposal

```
sql> desc tgap_face;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 face_id                                            number
 imp_low                                            number
 imp_high                                           number

sql>  desc tgap_edge; /* note: all versions of edge in single record */
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 edge_id                                            number
 imp_low                                            number
 imp_highs                                          varray(number)
 start_node_id                                      number
 end_node_id                                        number
 left_face_ids                                      varray(number)
 right_face_ids                                     varray(number)
 blg_id                                             number

sql> desc tgap_blg_original;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 blg_id                                    not null number(11)
 tree_source                                        blgtree

sql> desc tgap_blg_joided;
 name                                      null?    type
 ----------------------------------------- -------- ---------------------
 blg_id                                    not null number(11)
 child1_id                                          number(11)
 child2_id                                          number(11)
 top_node                                           number(11)
 top_tolerance                                      float(126)

sql> desc tgap_node;
 name                                      null?    type
 ----------------------------------------- --------
 node_id                                            number
 geometry                                           mdsys.sdo_geometry

sql> create view tgap_blg as (select
            blg_id,
            tree_source,
            null as child1_id,
            null as child2_id,
            null as top_node,
            null as top_tolerance
        from tgap_blg_original)
    union all (select
            blg_id,
            null as tree_source,
            child1_id,
            child2_id,
            top_node
            top_tolerance
        from tgap_blg_joined);
```