Modelling Higher Dimensional Data for GIS Using Generalised Maps*

Ken Arroyo Ohori Hugo Ledoux Jantien Stoter

May 2, 2013

Abstract

Real-world phenomena have traditionally been modelled in 2D/3D GIS. However, powerful insights can be gained by integrating additional non-spatial dimensions, such as time and scale. While this integration to form higher-dimensional objects is theoretically sound, its implementation is problematic since the data models used in GIS are not appropriate. In this paper, we present our research on one possible data model/structure to represent higher-dimensional GIS datasets: generalised maps. It is formally defined, but is not directly applicable for the specific needs of GIS data, e.g. support for geometry, overlapping and disconnected regions, holes, complex handling of attributes, etc. We review the properties of generalised maps, discuss needs to be modified for higher-dimensional GIS, and describe the modifications and extensions that we have made to generalised maps. We conclude with where this research fits within our long term goal of a higher dimensional GIS, and present an outlook on future research.

1 Introduction

Spatial data modelling refers to the creation of abstract mathematical representations of real world objects embedded in space. This includes not only purely spatial aspects, such as the objects' geometry and topology, but also other characteristics required for their use, such as the ability to attach attributes (both for storage and for thematic aspects), mark visited objects, or to have efficient access to the objects within a region.

Spatial models have been developed largely independently in the disciplines that required information on spatial objects, including computer graphics, computer-aided design and manufacturing (CAD/CAM), geology, and geographic information systems (GIS) [1]. Because of their independent creation, they are a reflection of the idiosyncrasies of their domains and differ significantly in key

*Author's draft of the paper to be presented at the 13th International Conference on Computational Science and Its Applications, within the session Spatial Data Structures and Algorithms for Geoinformatics, held in Ho Chi Minh City, Vietnam.

issues. One consequence of this is that in many fields support for 3D data has been long widespread and the theoretical foundations for higher dimensions are well established. However, GIS still has limited support for 3D data, and higher-dimensional GIS, despite decades of frequent mentions in literature [2, 3, 4, 5], remains in most cases a theoretical discussion[1].

This slow progress in the GIS world is not due to a lack of applications in higher dimensions. While being limited to 3D space is acceptable to many users of geographic information, substantial work has been done regarding the integration of non spatial-dimensions [6], such as time [7, 8] and scale [9, 10], to spatial data models. This is done either by creating specific models for these non-spatial dimensions, or by treating them as additional spatial ones [5], yielding a *higher dimensional spatial model*. The latter case is more extensible and generic, allowing us to manipulate objects in a *dimension independent* manner [11]. It is also the focus of this paper, and therefore the notion of a higher dimensional spatial model is first explained in detail in Section 2.

Since there is both a need for higher-dimensional GIS, and an availability of such data models from other fields that are able to support higher-dimensional data, there is great potential in finding a suitable model and adapting it to the specific needs of real-world (GIS) data, such as: support for overlapping regions, holes, and complex handling of attributes and metadata; and providing the specific operations that are required for its use, such as good construction and querying operations, buffering and overlays [12]. A short summary of the most remarkable representations for higher dimensional objects developed in other fields and that could thus be adopted is given in Section 3.

Among these, we propose the use of *generalised maps*, which are explained in Section 4, a model capable of representing a wide class of objects in arbitrary dimensions. It has several advantageous properties, such as support for unbounded objects (useful for time and other unbounded dimensions [13]), avoiding problems with incompatible orientations (a common problem when objects are built independently), and providing a simple manner to attach attributes to the objects of every dimension (e.g. vertex, edge, facet, etc.). Practically, it also has the advantage of having been implemented in 3D (it is used in GOCAD[2] for geological modelling and in Moka[3] for geometric modelling).

However, generalised maps by themselves cannot support all the characteristics of real-world spatial data. To bring our ideas into practice, in Section 5 we therefore explain how we have modified and implemented generalised maps for this purpose, and how some specific challenging aspects of GIS data can be handled. We finalise with our conclusions, discussion and our plans for future work in Section 6.

[1] Among the implementations that do exist, this term is most often used as a catchphrase for any processing involving 3D space and time. However, time is usually treated as a mere attribute, and true 4D space is almost never used.

[2] `http://www.gocad.org/`

[3] `http://moka-modeller.sourceforge.net/`

2 Higher-dimensional spatial information

The simplest technique to handle additional dimensions in spatial information, both in GIS and other fields, is using multiple independent representations. In practice, this means that these dimensions are considered as simple attributes which are attached to 2D or 3D objects. Such is the case in so-called 2.5D models for height, the 'snapshot' model for time [14] and most approaches to multi-scale data, including CityGML [15]. This approach is simple to understand and implement, but it also has important disadvantages:

- There is only a fixed (discrete) number of representations, which means that the objects being represented only have a known value at certain predefined points along the dimension. For example, a moving object's position is only known at certain moments.
- There is no link between the same object at different representations, which makes it difficult to maintain consistent representations after updates and precludes topological queries along this dimension. For instance, finding a moving object involves a brute force search, and checking if two objects at different scales are equivalent can only be inferred indirectly.
- The geometric and topological information is stored multiple times, which is wasteful in memory and can easily lead to changes being propagated incorrectly (or not at all), resulting in inconsistencies.

Many other approaches add some topology and additional information to these independent 2D or 3D representations. For instance, event-based models [16] connect successive moments in time with the changes that occurred in them, object-relationship models [17] add information to model the changes themselves, and the original *tGAP* structure [18] links appropriate 2D objects at different levels of detail. These representations are lightweight and sufficient for many applications, but they do not solve any of the above mentioned problems in their entirety: there is still a fixed number of points along a dimension (e.g. levels of detail or moments in time), some topological relations are not possible to keep in an efficient manner (especially along the additional dimensions), and inconsistencies are easy to generate when combining data sources or manipulating objects without special care.

Because of this, others have proposed to treat all dimensions as spatial ones (see [19] for time and [20] for scale). This solution is more complex, but it means that objects have known geometry, topology and attributes at all possible values within a range. Alternatively, this can be seen as having access to all the topological relationships between the objects, down to the vertex-to-vertex level. This helps to avoid redundancies and inconsistencies in the data. What we mean by treating all dimensions as spatial ones is explained as follows.

For simplicity, let us first consider a case with 2D space, time as the third dimension, and only linear (flat) geometries. At any one point in time, an object would be represented as a polygon in 3D space, and it would be parallel to the

2D space plane and orthogonal to the time axis. Every object existing (and not moving or changing shape) during a time period would then be a prism, with its base and top parallel to the 2D space plane and the other facets orthogonal to it. An example of this situation is shown in Figure 1.

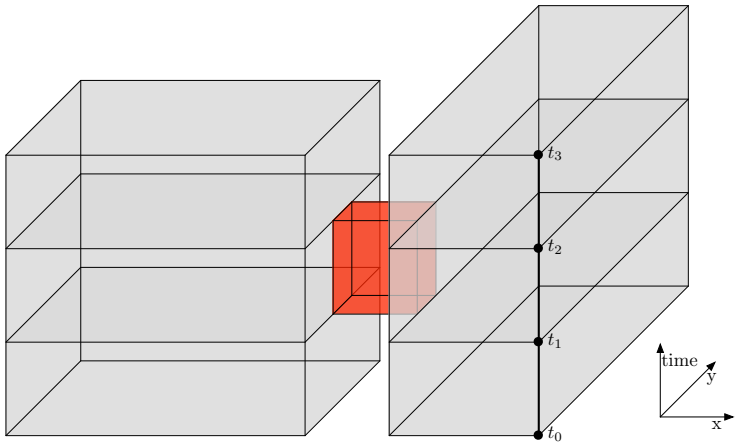


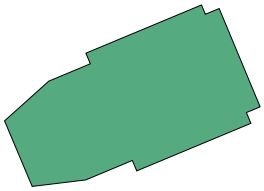
Figure 1: A 2D space (x, y) + time (vertical axis) perspective view of the footprint of two separate buildings at time t_0, which were connected by a corridor (red) from time t_1 to time t_2 and then became disconnected again when the corridor was removed until time t_3. The moments in time are shown along the thick line representing the front right corner of the right building.

Extending this to a 4D representation of 3D space and time, every object at one point in time would be a polyhedron in 4D space, and an object that exists for a period of time would be a polychoron, i.e. the four-dimensional analogue of a polygon/polyhedron. If this object is not moving or changing shape, it would take the form of a prismatic polychoron, i.e. the four-dimensional analogue of a prism. A simple example of such an object, generated by successive extrusions of a 2D footprint from the GBKN[4] data set, is shown in Figure 2.

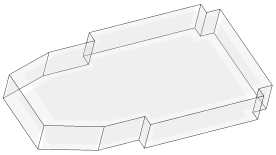
3 Higher-dimensional data models

There are several data models that are able to support higher-dimensional objects. However, most of these are limited to point or raster data, which have trivial or no topology, and are thus much more straightforward to use and implement. Higher-dimensional point clouds are common in data mining [21], while higher-dimensional rasters are common in medical imaging [22], among other examples.

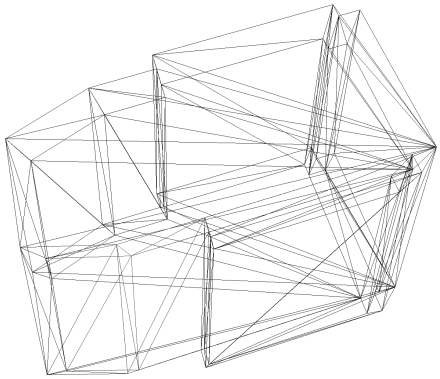
[4] `http://www.gbkn.nl`, a Dutch large-scale topographic data set



(a) The GBKN footprint of the Aula Congress Centre in Delft



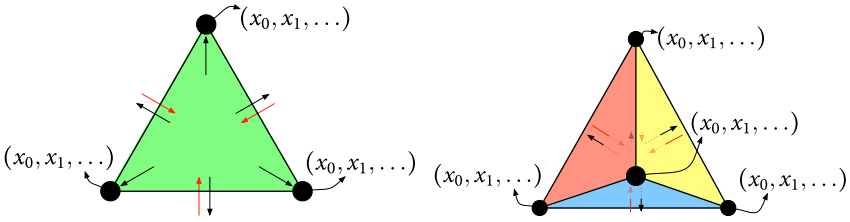
(b) After extruding it to create a block shaped polyhedron (perspective projection of edges and facets only)



(c) After extruding it again (double perspective projection of the edges only)

Figure 2: A 4D (3D+time) representation of the Aula Congress Centre in Delft.

For vector data consisting of closed polytopes (i.e. the higher-dimensional analogue of a polygon/polyhedron), there are fewer options. A deceptively simple one involves the geometric subdivision of an n-dimensional polytope into n-dimensional simplices, i.e. an n-dimensional simplicial decomposition or n-dimensional triangulation, which can be easily represented and stored using an incidence model, as shown in Figure 3. In its simplest form, a data structure for this could be a list of vertex coordinates, and a list of simplices, each containing the $n+1$ vertices that define it and the $n+1$ simplices that are adjacent to it. This option has several advantages: the data structures are simple to use and implement, it can be compressed with relative ease [23, 24], and operations between simplices are much more straightforward than those between arbitrary polytopes (e.g. the intersection of two simplices of a certain dimension can yield only a limited number of different configurations).



(a) Triangle based representation (b) Tetrahedron-based representation

Figure 3: n-simplex based data structures in 2D and 3D. Black arrows represent pointers from the element shown, while red arrows show the ones from other elements that point to this one.

However, doing this subdivision is extremely difficult in practice, since it requires the creation of an n-dimensional constrained triangulator, which has been described in theory [25, 26] for some cases, but has never been implemented. Doing so for the general case, especially in a robust manner, would prove very difficult.

Another option, and the one further discussed in this paper, is using *ordered topological models* [27], a type of boundary representations that are based on a single type of fundamental constructing element (e.g. a half-edge), on which a usually small number of pre-defined functions act. The more complex elements and connected components of such a model are only defined implicitly, e.g. as a set of fundamental elements. This allows objects to be represented as is, at least from a high level perspective, not needing to conform to a particular shape (unlike decomposition models like rasters).

Such data models also have the advantage of separating the topology of the objects (which is dealt with in the model directly), and their geometry (which is dealt with in an embedding model). This is a useful property, since it distinguishes the problems in geometric modelling from those in computational geometry [28]. Algorithms and methods from both fields can then be applied indistinctly to solve specific problems.

There are other possible models which are not discussed further in this paper but still represent interesting possibilities. Constructive models based on constructive solid geometry, alternate decompositions [29] or intersections of half-spaces have a strong theoretical background, but attaching attributes to individual elements is difficult, and realising an object involves complex geometric computations. Nef polyhedra [30] are very powerful, but require the construction of an n-dimensional hyperspherical projective kernel, which is also a very complex task in practice [31].

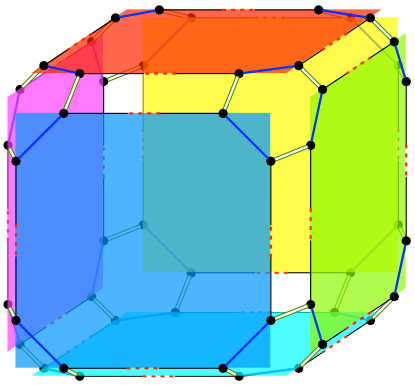
4 Generalised maps

Generalised maps (sometimes shortened as G-maps) are an ordered topological model developed by Liendhardt [32] based on the concept of a combinatorial map, also known as a topological map, which was described by Edmonds [33]. They are roughly equivalent to the cell-tuple structure of Brisson [34], but were shown to be able to represent the topology of a wider class of objects, i.e. orientable or non-orientable cellular quasi-manifolds with or without boundary—manifolds partitioned into cells [35] that allow certain types of singularities, as long as every i-dimensional cell (i-cell) is incident to no more than two $(i+1)$-cells in a $(i+2)$-cell.

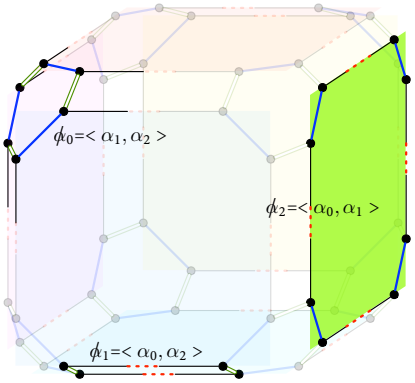
A generalised map is composed of two elements: *darts* and *involutions* (α). The precise definition of a dart is complex [32], but since for our application we are only interested in representing linear (flat) geometries, a dart can be intuitively seen as a unique combination of a specific *i-cell* (a vertex being a 0-cell, an edge is a 1-cell, a facet is a 2-cell, and so on) in each dimension, all of which are incident to each other. Meanwhile, involutions are bijective operators connecting darts that are related along a certain dimension. In this manner, α_0 joins darts into edges, α_1 connects consecutive edges within a facet, α_2 connects adjacent facets within a volume, and so on. An example of a 3D generalised map (3-G-map) representation of two adjacent cubes is shown in Figure 4, where α_0 thus joins vertices to form edges, α_1 connects consecutive edges within a facet, α_2 connects adjacent facets within a volume, and so on.

One can traverse the combinatorial structure by the use of the *orbit* operator, which returns a set of darts that are reachable by following certain involutions only. To obtain the darts that are part of a certain i-cell only, one can start from any dart d belonging to the i-cell, following all involutions *except* for α_i. This is commonly denoted an $<\not\alpha_i>$ orbit of d [36]. Since α_i connects *adjacent* i-cells, not following it means staying within the same i-cell. For simple construction, the *sew* operation is used, connecting two objects of the same dimension along the common face, i.e. $(i-1)$-cell, in their boundaries. Analogously, the *unsew* operation can be used to unset these involutions.

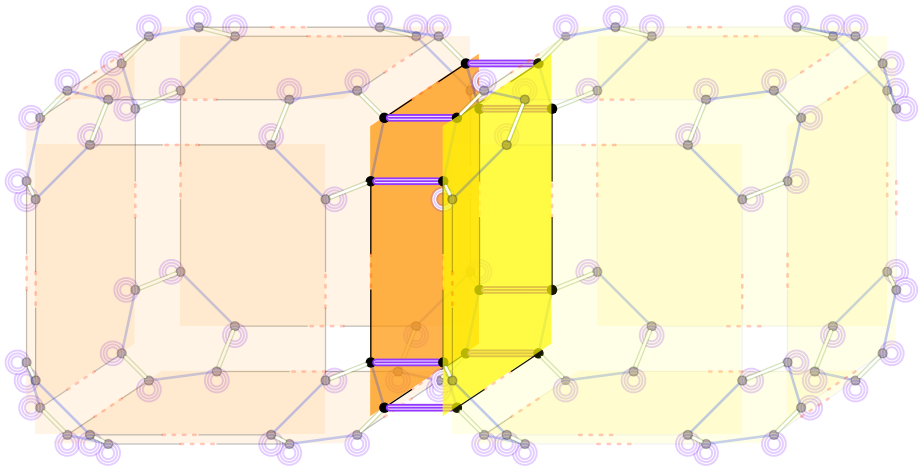
More formally, a n-dimensional generalised map is defined by a $(n+2)$-tuple $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$, where D is a non-empty set of darts, and α_i is an involution (i.e. $\forall d \in D, \forall 0 \leq i \leq n, \alpha_i(\alpha_i(d)) = d$) that connects objects of dimension i, and $\forall 0 \leq i \leq n-2, \forall i+2 \leq j \leq n, \alpha_i(\alpha_j(d))$ is also an involution.



(a) A G-map representation of a cube.



(b) The $\not\alpha_i$ operator obtains all the darts belonging to a specific i-cell. Thus, $\not\alpha_0$ obtains the darts belonging to a vertex, $\not\alpha_1$ those belonging to an edge, and $\not\alpha_2$ those belonging to a facet.



(c) A G-map representation of two cubes. Note how the individual cubes have identical involutions to those of (a), with the addition of an α_3 involution that connects the two cubes at their common face. In the other darts, this involution is not used.

Figure 4: A 3D G-map representation of a pair of adjacent cubes, showing the α_0 (dashed red), α_1 (solid blue), α_2 (double green), α_3 (triple purple), and $\not\alpha_i$ operators.

In order to traverse a G-map, the orbit operator $< A > (d) =< \alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{in} > (d)$ obtains all the darts that can be reached from dart d by successive applications of the operators $\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{in} \in A$. For convenience, the operator $< \not{\alpha_i} > (d)$ is defined as well, which traverses all α involutions *except* for α_i [36], obtaining all the darts that are part of the same i-cell as d.

The construction of objects in its simplest form is based on the sewing operator, which joins two i-cells along the $(i-1)$-cell that lies in their geometric common boundary. Thus, it takes two corresponding darts d_1 and d_2 ($\forall 0 \leq j \leq n, i \neq j \iff$ d_1 and d_2 belong to the same j-cell) on opposite sides of the common $(i-1)$-cell, computes their $< \alpha_0, \alpha_1, \ldots, \alpha_{i-1} > (d_1)$ and $< \alpha_0, \alpha_1, \ldots, \alpha_{i-1} > (d_2)$ orbits, performs a parallel traversal of both, and connects them by adding α_i involutions that connect the corresponding darts from each orbit along the i-th dimension. Note that this implies the use of consistent ordering criteria in the orbit operator, such as always following the lowest possible α involution first. The unsew operator similarly uses a single $< \alpha_0, \alpha_1, \ldots, \alpha_{i-1} > (d)$ orbit (since the $(i-1)$-cells are now linked) to remove all the involutions between any darts in the orbit along the i-th dimension.

5 Implementation

There are many possible realisations of generalised maps as a data structure. For instance, a minimal data structure that stores the combinatorial aspect of an n-G-map could involve a single type of object, a `Dart` with $n+1$ pointers to other darts representing its involutions. However, another option could be to have a set of `Involutions` that store the identifiers of the two darts that each of them link. These two options are presented in Figure 5.

```
struct Dart {
    Dart *involutions[n+1];
};
```

(a) Based on darts

```
struct Involution {
    id dart1, dart2;
};
```

(b) Based on involutions

Figure 5: A minimal G-maps implementation

Nevertheless, these data structures by themselves do not store any geometry or support many of the characteristics of GIS data. An implementation for use in GIS requires:

1. **Geometry and topology** Storing not only topological relationships, but also the geometry of the objects. At least linear (flat) objects should be supported.

2. **Attributes** Storing complex attributes of different types (e.g. numeric, text, an element of a discrete set of classes, etc.), possibly at every dimension (e.g. vertex, edge, face, etc.). Every i-cell can have a tuple of

attributes of different types, but all the cells of a certain dimension generally have the same attribute types in their tuples.

3. **Construction** Constructing a model from both topological or non topological data. Topological data might need to be checked (in case the topological information does not match the actual geometry of the objects), while non topological construction should be performed in a consistent manner, generating valid topological information and ensuring that objects that are geometrically equivalent are only generated once.

4. **Queries** Answering geometric, topological and attribute based queries efficiently. In order to do this, all necessary links between the objects should be kept, and an external data structure for spatial indexing might be required as well.

5. **Holes** Storing and efficiently accessing void regions in possibly every dimension higher than 0. To ensure a consistent model, these holes should fit inside their containing object, which implies that they should be of the same dimensionality or lower.

6. **Disconnected and overlapping objects** Keeping track and traversing objects even when they form topologically disconnected groups. They might be disconnected by virtue of being geometrically disconnected, or also by being in a configuration that is not directly representable using generalised maps. This implies that a higher level structure that somehow maintains this information is required. This data structure can however have many possible forms.

The data structures presented previously are only sufficient to represent the combinatorial structure (**topology**) of a generalised map, equivalent to the topological relationships in a partition of space without holes. However, to represent the geometry and other characteristics of the model, some modifications and additional structures are needed. These are shown in Figure 6 and explained as follows.

To store **geometry**, *embedding* structures are used; each one of these containing the geometry of a specific i-cell. Since only linear geometries are required, only the 0-dimensional point embeddings are strictly necessary, which store the coordinates of each vertex. The geometry of the higher-dimensional embeddings can then be inferred from the points in their boundary. Since each dart represents a unique combination of an i-cell of each dimension, a dart can be linked to its corresponding embedding structure for each dimension. On the other direction, it is sufficient to link an embedding to any one dart representing part of its boundary.

Attributes and **holes** work in a similar manner. Since the tuples of attribute types of all i-cells (cells of equal dimension) are equal, one embedding data structure per dimension storing the attributes of that dimension, is sufficient. A list of holes present in that i-cell can be then kept as an additional attribute, its only practical requirement being that the dimensionality of the

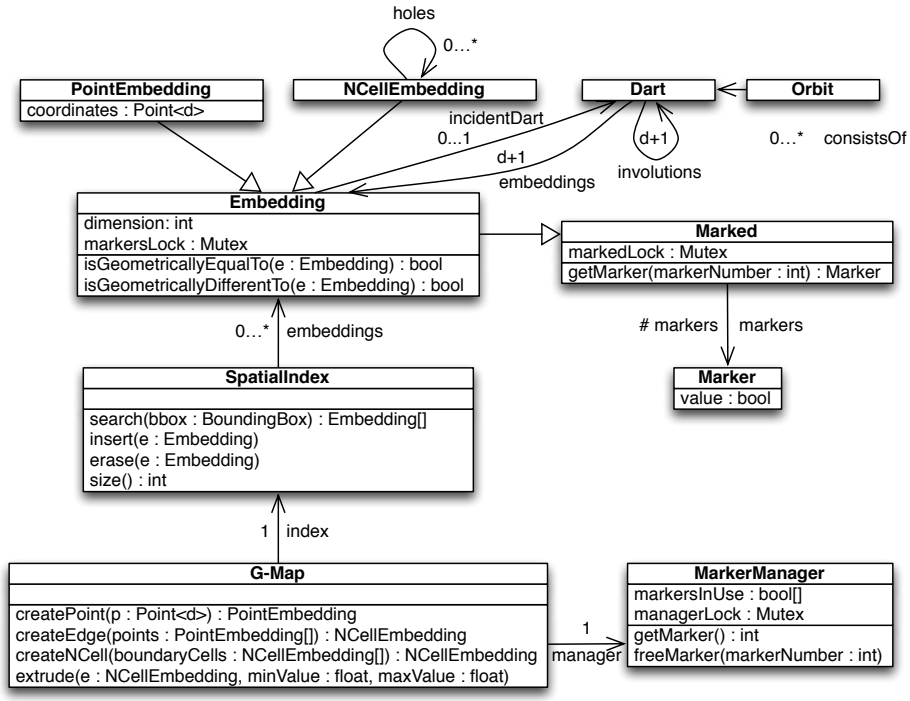


Figure 6: Our implementation of generalised maps for real-world GIS data.

hole (represented as an embedded cell as well) should be equal or lower than that of the containing cell, and that its geometry should be fully inside the containing cell.

Meanwhile, **queries** and **disconnected objects** are handled through the use of a spatial index. For this purpose we have investigated several options, among which the most promising options are R-tree variants like the R*-tree, or a simple index using a single vertex per cell, such as the lexicographically smallest one. Usual R-tree implementations are not practical since they have problems when dealing with objects of heterogeneous dimension (e.g. an object with zero-length along a particular dimension has a volume[5] of zero in higher dimensional space). The most important aspect of such a spatial index is that it allows us to maintain a connected graph.

Finally, we have developed two **construction** operators: a higher dimensional analogue of extrusion, which uses a d-dimensional object and a range along the $(d+1)$-th dimension to create a $(d+1)$-dimensional object; and an incremental construction methods that creates a $(d+1)$-dimensional object described by the d-dimensional objects in its boundary. These are the focus of two upcoming articles and are not discussed here any further.

6 Conclusion and Future Work

This paper presents part of our ongoing research to implement a higher-dimensional GIS based on mathematical models that have been developed in other domains. This will enable the full integration of the separate dimensional aspects of GIS, such as 2D/3D space, time and scale [37]. We have modified and extended generalised maps into a data structure that is not much more complex than a basic implementation, but one that is able to support the real-world characteristics that are found in GIS data.

Our future work will cover: the visualisation of higher-dimensional data, efficient construction techniques, improved spatial indexing, keeping the consistency and validity of data, and improving the memory consumption of generalised maps.

Acknowledgements

This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs (Project code: 11300).

References

[1] Frank, A.U.: Spatial concepts, geometric data models, and geometric data structures. Computers & Geosciences **18**(4) (1992) 409–417

[5] more precisely, a Lebesgue measure

[2] Hazelton, N., Leahy, F., Williamson, I.: On the design of temporally-referenced, 3-D geographical information systems: development of four-dimensional GIS. In: Proceedings of GIS/LIS'90. (1990)

[3] Hansen, H.S.: A quasi-four dimensional database for the built environment. In: Proceedings of the 1st International Symposium on Digital Earth Moving. Volume 2181 of Lecture Notes in Computer Science. (2001) 48–59

[4] O'Conaill, M.A., Bell, S.B.M., Mason, N.C.: Developing a prototype 4D GIS on a transputer array. ITC Journal (1) (1992) 47–54

[5] Raper, J.: Multidimensional geographic information science. Taylor & Francis (2000)

[6] Worboys, M.F.: A unified model for spatial and temporal information. The Computer Journal **37**(1) (1994) 26–34

[7] Goodchild, M.F.: Geographical data modeling. Computers & Geosciences **18**(4) (1992) 401–408

[8] Peuquet, D.J.: Representations of Space and Time. Guilford Press (2002)

[9] van Oosterom, P., Meijers, M.: Towards a true vario-scale structure supporting smooth-zoom. In: Proceedings of the 14th ICA/ISPRS Workshop on Generalisation and Multiple Representation, Paris. (2011)

[10] Li, Z.: Reality in time-scale systems and cartographic representation. The Cartographic Journal **31**(1) (1994) 50–51

[11] Karimipour, F., Delavar, M.R., Frank, A.U.: A simplex-based approach to implement dimension independent spatial analyses. Computers & Geosciences **36**(9) (September 2010) 1123–1134

[12] Albrecht, J.: Universal Analytical GIS Operations. A Task-Oriented Systematization of Data Structure-Independent GIS Functionality Leading Towards a Geographic Modeling Language. PhD thesis, University of Vechta (1995)

[13] Thompson, R.J., van Oosterom, P.: Integrated representation of (potentially unbounded) 2D and 3D spatial objects for rigorously correct query and manipulation. In Kolbe, T.H., König, G., Nagel, C., eds.: Advances in 3D Geo-Information Sciences. Lecture Notes in Geoinformation and Cartography. Springer (2011) 179–196

[14] Basoglu, U., Morrison, J.: The efficient hierarchical data structure for the US historical boundary file. In Dutton, G., ed.: Harvard Papers on Geographic Information Systems. Volume 4. Addison-Wesley (1978)

[15] OGC: OpenGIS City Geography Markup Language (CityGML) Encoding Standard. Open Geospatial Consortium. 1.0.0 edn. (August 2008)

[16] Peuquet, D.J., Duan, N.: An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data. International Journal of Geographical Information Science **9**(1) (1995) 7–24

[17] Claramunt, C., Parent, C., Spaccapietra, S., Thériault, M.: Database modelling for environmental and land use changes. In Stillwell, J.C.H., Geertman, S., Openshaw, S., eds.: Geographical Information and Planning. Advances in Spatial Science. Springer Berlin / Heidelberg (1999) 181–202

[18] van Oosterom, P.: 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) (2005) 331–346

[19] Tøssebro, E., Nygård, M.: Representing topological relationships for spatiotemporal objects. Geoinformatica **15** (2011) 633–661

[20] van Oosterom, P., Meijers, M.: Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data. In: Jaarverslag 2011. Nederlandse Commissie voor Geodesie (2012)

[21] Casali, A., Cicchetti, R., Lakhal, L.: Cube lattices: a framework for multidimensional data mining. In: Proceedings of the 3rd SIAM International Conference on Data Mining. (2003) 304–308

[22] McInerney, T., Terzopoulos, D.: A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis. Computerized Medical Imaging and Graphics **19**(1) (1995) 69–83

[23] Snoeyink, J., van Kreveld, M.: Good orders for incremental (re)construction. In: Proceedings of the 13th ACM Symposium on Computational Geometry. (1997) 400–402

[24] Blandford, D.K., Blelloch, G.E., Cardoze, D.E., Kadow, C.: Compact representations of simplicial meshes in two and three dimensions. International Journal of Computational Geometry and Applications **15**(1) (2005) 3–24

[25] Shewchuk, J.R.: Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. In: Proceedings of the 16th Annual Symposium on Computational Geometry. (2000) 350–359

[26] Shewchuk, J.R.: General-dimensional constrained Delaunay and constrained regular triangulations, I: Combinatorial properties. Discrete & Computational Geometry **39**(1003) (March 2008) 580–637

[27] Lienhardt, P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. Computer-Aided Design **23**(1) (1991) 59–82

[28] Mäntylä, M.: An introduction to solid modeling. Computer Science Press, New York, USA (1988)

[29] Bulbul, R., Frank, A.U.: AHD: The alternate simplicial decomposition of nonconvex polytopes (generalization of a convex polytope based spatial data model). In: Proceedings of the 17th International Conference on Geoinformatics. (2009) 1–6

[30] Bieri, H., Nef, W.: Elementary set operations with d-dimensional polyhedra. In: Computational Geometry and its Applications. Volume 333 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1988) 97–112

[31] Granados, M., Hachenberger, P., Hert, S., Kettner, L., Mehlhorn, K., Seel, M.: Boolean operations on 3D selective nef complexes: Data structure, algorithms and implementation. In: Proceedings of the 11th Annual European Symposium on Algorithms. (September 2003) 174–186

[32] Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. International Journal of Computational Geometry and Applications **4**(3) (1994) 275–324

[33] Edmonds, J.: A combinatorial representation of polyhedral surfaces. Notices of the American Mathematical Society **7** (1960)

[34] Brisson, E.: Representing geometric structures in d dimensions: topology and order. In: Proceedings of the 5th annual symposium on Computational geometry, New York, NY, USA, ACM (1989) 218–227

[35] Hatcher, A.: Algebraic Topology. Cambridge University Press (2002)

[36] Lévy, B., Mallet, J.L.: Cellular modeling in arbitrary dimension using generalized maps. Technical report, ISA-GOCAD (1999)

[37] van Oosterom, P., Stoter, J.: 5D data modelling: Full integration of 2D/3D space, time and scale dimensions. In: Proceedings of the 6th International Conference GIScience 2010, Springer Berlin / Heidelberg (2010) 311–324