Representing the dual of objects in a four-dimensional GIS[*]

Ken Arroyo Ohori   Pawel Boguslawski   Hugo Ledoux

Abstract

The concept of duality is used to understand and characterise how geographical objects are spatially related. It has been used extensively in 2D to qualify the boundaries between different types of terrain, and in 3D for navigation inside buildings, among others. In this paper, we explore duality in four dimensions, in the context where space and other characteristics (e.g. time) are modelled as being in four dimensional space. We explain what duality in 4D entails, and we present two data structures that can be used to store the dual graph of a set of 4D objects. We also discuss applications where such data structures could be useful in the future.

1   Introduction

The concept of duality is used in geographical information systems (GIS) to understand and represent how things are connected, and to characterise spatial relationships. In two dimensions, one application is qualifying the spatial relationships between adjacent objects: as shown in Figure 1, Gold (1991) uses two connected data structures to store simultaneously a polygonal map (where each polygon has certain attributes) and its dual (the boundaries between two map objects having certain attributes, e.g. the boundary type or the flow direction). He argues that the boundaries do not characterise *per se* any of the objects, but rather the adjacency relationships that exist between them. The data structure used was the *quad-edge* structure of Guibas and Stolfi (1985). In three dimensions, duality also permits us to understand how different solids are spatially related (e.g. two rooms in a building are *adjacent*). Arguably the most known use of the dual is to model navigational paths inside three-dimensional buildings. Lee and Zlatanova (2008) and Lee and Kwan (2005) extract from a 3D building a graph that can be used in case of emergency, and Boguslawski et al. (2011) and Boguslawski (2011) perform the same using a data structure, the *dual half-edge* (DHE), which simultaneously represents the buildings (the rooms and their boundaries) and the navigation
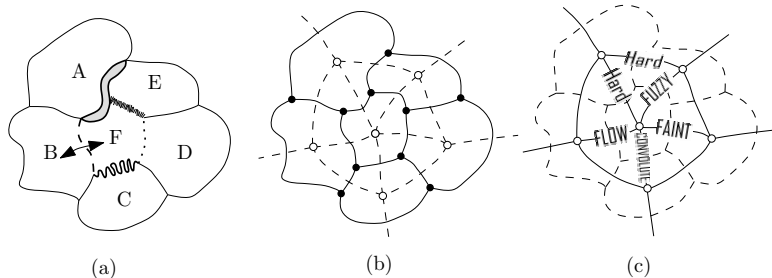

Figure 1: **(a)** Six map objects and their boundaries. **(b)** The same map stored as a graph and its dual (dotted lines). **(c)** The dual graph is used to describe the relationships between adjacent polygons. (Figure after Gold (1991))

graph. With the DHE, the construction and manipulation operations update both representations at the same time, permitting the simultaneous modelling and characterisation of buildings. There are several other examples of duality in GIS: the Delaunay triangulation and the Voronoi diagram are often used to model continuous phenomena, these two structures being dual to each other. Dakowicz and Gold (2003) use them for terrain modelling, Lee and Gahegan (2002) for interactive analysis, and Ledoux and Gold (2008) for three-dimensional fields in geosciences.

In this paper, we are interested in the concept of duality in four-dimensional space to model four dimensional objects. These objects are the result of the integration of a non-spatial dimension to the three dimensions of space, to create 4D objects where all dimensions are treated as spatial (Raper, 2000). Examples of the non-spatial dimensions that can be used are: time (Peuquet, 2002; Worboys, 1994), scale (van Oosterom and Meijers, 2011; Li, 1994) and attributes (beyond 2.5D-type modelling). As van Oosterom and Stoter (2010) argue, the main advantage of such an integration is the *consistency* of data, both across space and the other dimensions modelled—with the proper validation functions, one can ensure that all the data for a given region is consistent across time or across different scales, for instance.

Adding an extra dimension implies that a 4D primitive has to be modelled: the *polychoron*[1], which is the 4D analogue of a polygon or polyhedron. To understand and characterise the spatial relationships between polychora (and between these and the lower-dimensionality primitives from which they are built), the dual graph of a set of polychora can be constructed and analysed. For example, in a 3D model of a building where rooms are represented by polyhora we could locate people inside the building. With the dual graph, a user would be able to know where in 3D space a given user was at any time, when this person moved from a given room to another one, or the shortest path between any two rooms at any given time.

We first describe in Section 2 our terminology, the kinds of objects

---

[1]Also called a 4-polytope or a 4-polyhedron.

we are modelling in 4D, and the concept of duality. We then present in Section 3 potential data structures to store the dual graph of a partitioning of a 4D space. We first describe how *generalised maps* (Lienhardt, 1994) can be used to extract and store the graph, and then we discuss how the DHE (Boguslawski et al., 2011; Boguslawski, 2011) can be modified to store simultaneously both the partitioning and the dual in 4D. We analyse in Section 4 the storage of each structure, as one of our aim is to efficiently implement a 4D structure where the dual is available, and we briefly discuss in Section 5 potential applications of such data structures.

2 The four-dimensional Euclidean space, and duality

If we denote the three axes of the three spatial dimensions by x, y and z, then the axis w of the fourth dimension is perpendicular to that of all the spatial dimensions. In 4D Euclidean space (denoted $\mathbb{R}^4$), the simplest 4D primitive, called a *simplex*, is a 5-vertex polychoron and it is analogous to the triangle in 2D and to the tetrahedron in 3D. More generally, a d-dimensional simplex, also denoted as a d-simplex, is the convex hull of a set of $(d+1)$ linearly-independent points in $\mathbb{R}^d$. Constructing a 4-simplex can be performed hierarchically: from a tetrahedron, we first embed its 4 vertices in 4D (with four coordinates each), then one new vertex is added, and finally 4 new edges must be constructed (these join the new vertex to the existing 4 of the tetrahedron). The resulting polychoron has 5 vertices, 10 edges, 10 triangular faces (2-simplices), and its boundary is formed by 5 tetrahedra (3-simplices).

By integrating 3D space and the extra dimension into 4D space, we ensure that there are no gaps or overlaps. This implies that we create a partitioning of $\mathbb{R}^4$. This can be achieved by keeping a so-called "universe polychoron" which encloses all the other polychora present in the model, in a manner similar to Liu and Snoeyink (2005). The structure we create is thus a partitioning into *cells*, where a 0-cell is a vertex, a 1-cell an edge, a 2-cell a polygon, a 3-cell a polyhedron, and a 4-cell a polychoron. Currently, no holes inside cells are allowed in our definition. We name a $(k-1)$-cell incident to a k-cell a *facet* of it; a facet of a 4-cell is thus a 3-cell that lies in its boundary. This resulting partitioning forms a *cell complex* C, which is a finite set of cells having the following two conditions:

1. A facet of a k-cell in C is also in C;

2. The intersection of two cells σ_1 and σ_2 in C, denoted $\sigma_1 \cap \sigma_2$, is either empty or is in C.

A cell complex in $\mathbb{R}^4$ can be represented by a graph where the vertices and edges are embedded in $\mathbb{R}^4$ such that a set of vertices and edges implicitly represent a cell. Observe that since there are no holes allowed, the graph is connected.

Duality can have many different meanings in mathematics, but it always refers to the translation or mapping in a one-to-one fashion of concepts or structures. We use it here in the sense of the dual of a given
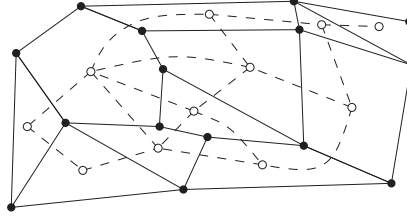
Figure 2: A graph G (solid lines), and its dual graph $G^\star$ (dashed lines). For the sake of simplicity the dual edges to the edges on the boundary of G are not drawn.
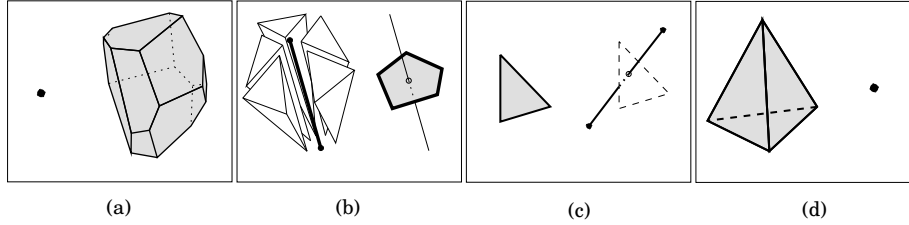

| (a) | (b) | (c) | (d) |

Figure 3: Duality in a 3D cell complex.

graph. Let G be a planar graph (thus embedded in $\mathbb{R}^2$), as illustrated in Figure 2 (solid edges); observe that G can also be seen as a cell complex in $\mathbb{R}^2$. The duality mapping is defined as follows. The dual graph $G^\star$ has a vertex for each face (polygon) in G, and these vertices are linked by an edge if and only if their two corresponding dual faces in G are adjacent (in Figure 2, $G^\star$ is represented with dashed lines). Notice also that each polygon in $G^\star$ corresponds to a vertex in G, and that each edge of $G^\star$ (arcs in Figure 2) is dual to an edge in G.

The concept of duality is valid in any dimension, as we consider a graph embedded in $\mathbb{R}^d$ as a d-dimensional cell complex. The mapping between the elements of a cell complex in $\mathbb{R}^d$ is simple: let C be a k-cell, the dual cell of C in $\mathbb{R}^d$ is denoted by $C^\star$ and is a $(d-k)$-cell. As a result, in four dimensions, a 0-cell becomes a 4-cell, and vice versa; a 1-cell becomes a 3-cell, and vice versa; and a 2-cell stays a 2-cell. Figure 3 shows the duality of a cell complex in $\mathbb{R}^3$.

3   Potential data structures

There are several data structures that are able to represent models in four or more dimensions. Notable ones include: simplex-based ones (Paoluzzi et al., 1993; Shewchuk, 2000), polytopal meshes (Sohanpanah, 1989), (convex) decompositions of polytopes (Bulbul et al., 2009), and Nef polyhedra (Bieri and Nef, 1988). However, despite the fact that they maintain various topological relationships, none of them provide efficient access to the dual graph of a model. We have nevertheless identified two candidate

data structures that are able to do so, generalised maps and the dual half-edge. In this section, these are first introduced in their general form, and afterwards we specifically analyse how they could handle the dual graph in four dimensions.

3.1 Dual half-edge

The dual half-edge (DHE) structure, as proposed by Boguslawski et al. (Boguslawski et al., 2011; Boguslawski, 2011), is a data structure that is able to represent a set of connected polyhedra forming a cell complex. It does so by simultaneously storing both the primal and the dual graphs of the objects, in a similar manner as the quad-edge structure of Guibas and Stolfi (1985) in 2D.

As shown in Figure 4a, with the DHE each polyhedron is represented

(a) The DHE models 3D subdivisions by representing the boundary of each polyhedron separately with a graph (edges are black lines), and two adjacent polyhedra are linked together by the dual graph (edges are dashed lines).

(b) The DHE pointer based data structure; the primal graph (solid lines) is connected permanently with the dual graph (dashed lines); he - original half-edge; S, N_V, N_F, D, V - pointers.
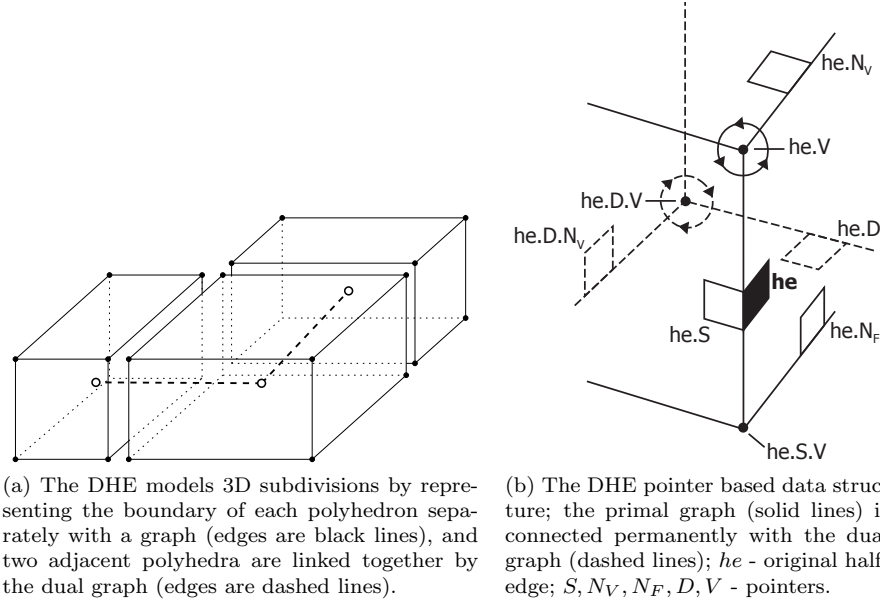
Figure 4: The dual half-edge data structure in 3D.

independently with an edge-based structure (a *b-rep* model), and adjacent polyhedra are linked together by their shared faces, which are represented by half-edges joining 3-cells. These form a graph of connections in the dual of the original (primal) graph. Both the primal and the dual graphs are identical in terms of structure (i.e. their basic elements and connections). Figure 4b shows an idea of the relationships that are stored for each edge. Since these graphs conform to the duality concept as explained in Section 2, the only cells that are needed to build a 3D model are the 0-cells (nodes) and 1-cells (edges); the nodes store the vertex coordinates, while the edges store the connections between the nodes. Meanwhile, the 2-cells (faces) and 3-cells (volumes) are only implicitly represented, but

their attributes can be stored in their dual counterparts, the 1-cells and 0-cells in the dual graph.

However, an edge is not an atomic element in the DHE. Each edge consists of two half-edges, each of them being permanently connected with its corresponding half-edge in the dual. This pair, half-edge in the primal graph and half-edge in the dual one, is called the *dual half-edge*, and forms the atomic element in this model. Each half-edge is represented with five pointers which keep references to: an associated vertex, the next edge around a shared vertex, the next edge around a shared face, the second half-edge of the edge, and to the dual half-edge.

These five pointers are necessary to represent complex models including non-manifold cases—when two cells are only linked by a shared vertex or edge. However, the number of pointers can be reduced by one if only cells linked by a shared face are taken into consideration. Additionally, a primal and dual half-edge pair can be merged and stored as a single record, since they are permanently connected—the number of pointers is reduced by one.

Using the data structure directly, without higher level construction operators would be extremely difficult—'manual' updating of pointers while edges are added to a model can easily cause many mistakes. Therefore, it is preferred to use the construction operators from Boguslawski (2011). They allow for model construction in an easy way, edge-by-edge, like in various CAD systems. Additionally, the dual graph is created automatically as the edges are added to the model and single cells are linked into a complex. These operators, used for modifications of the existing model, make only local changes in the primal and dual graph, and thus the whole dual graph does not need to be reconstructed after each modification.
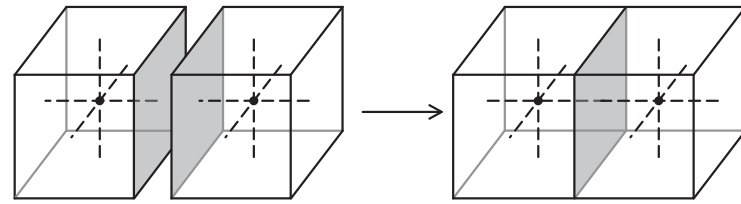
During the construction process, the external cell, which encloses cells in a complex representing a modelled object, is automatically created. It can be considered as 'the rest of the world'. This external cell prevents topological inconsistencies at the boundary of the complex, where cells do not have an adjacent cell to connect to. Also, navigation can be implemented without testing if a boundary of the complex is approached.

Figure 5 shows one possible way to construct two cubes linked into one complex. It is based on CAD-like operators—Euler operators (Baumgart, 1975; Braid et al., 1980; Mäntylä, 1988) and extended Euler operators (Masuda, 1993). First, two separate cubes are created (see Figure 5a). Then, they are linked by a shared face (see Figure 5b). It is possible to define different sequences which results in the same model. It should be noted that the external cell and dual graph are present at each step of the process, but for the sake of clarity the external cell and dual graph are not shown. The final model consists of three cells: two internal cubes and one external cell (see Figure 5c).

The DHE was originally designed for 3D models. However, a single polychoron can be represented using the DHE without any modifications, except for the use of 4D coordinates. This is done by instead representing the polyhedra that lie on its boundary, in a similar manner as a 2D data structure is commonly used to represent a single polyhedron by storing the polygons in its boundary, cf. Baumgart (1975). While this is not sufficient to represent a 4D cell complex with adjacent 4-cells or a non-
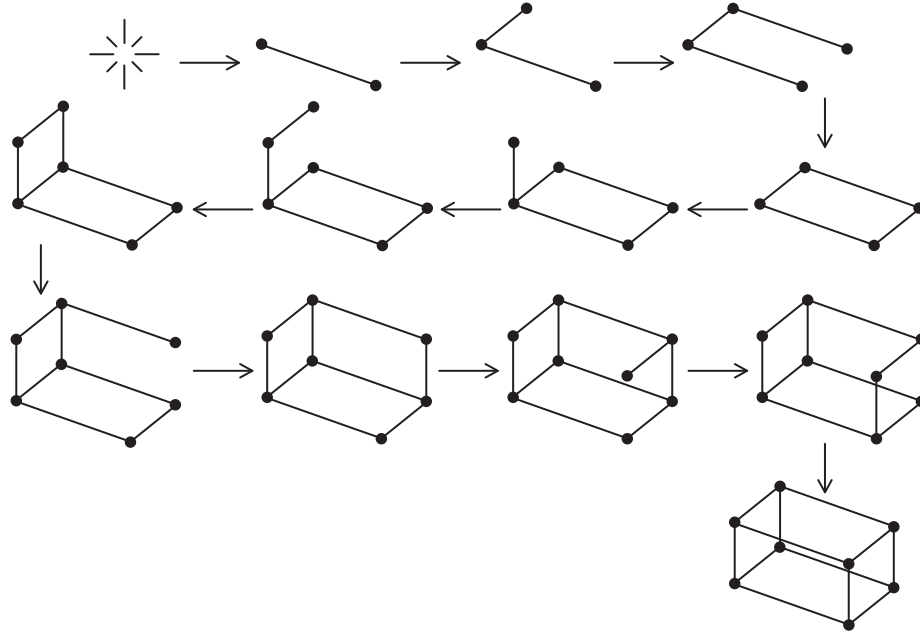
(a) A cube construction scenario.

(b) Cubes share a common face (grey). Cells in the complex are connected using dual edges (dotted lines).
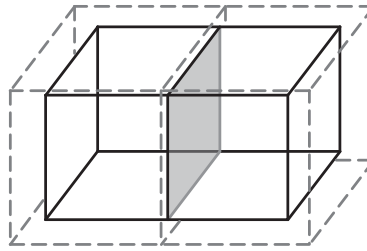
(c) The resulting model consists of two internal (solid lines) and one external (dotted grey lines) cells.

Figure 5: Cell complex construction process.

manifold object, we believe that the data structure can be extended to represent objects in 4D. The biggest challenge is to correctly manage all the connections between the 4-cells, so as to fulfil the 4D duality rules.

3.2   Generalised maps

Generalised maps (G-maps) are an ordered topological model developed by Lienhardt (1994) based on the concept of a combinatorial map, also known as a topological map, which was described by Edmonds (1960). They are roughly equivalent to the cell-tuple structure of Brisson (1989), but have been shown to be able to represent a wider class of objects known as cellular quasi-manifolds—manifolds that allow certain types of singularities, as long as every n-cell is incident to no more than two $(n+1)$-cells.

Intuitively, a G-map is composed of two elements: a set of *darts*, each of which is defined by a unique combination of a specific n-cell from every dimension, and are often represented visually as half-edges or oriented edges; and *involutions* (α), bijective operators connecting darts that are related along a certain dimension. In this manner, α_0 joins vertices to form edges, α_1 connects consecutive edges within a face, α_2 connects adjacent faces within a volume, and so on.

One can obtain the connected darts that form a specific cell by the use of the *orbit* operator, which returns a (possibly ordered) 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 as $< \not\alpha_i > (d)$ (Lévy and Mallet, 1999). Since α_i connects *adjacent* i-cells, not following it means remaining in the same cell. For simple construction, the *sew* operation is used, connecting two i-cells of the same dimension along the common part of their boundary. It does parallel traversals of two orbits, adding involutions that connect corresponding darts from each. Note that this implies certain ordering criteria in the orbit operator. Analogously, the *unsew* operation removes these involutions. An example of a 3D G-map representation of two adjacent cubes is shown in Figure 6.

The aforementioned elements and operations represent the combinatorial structure of a generalised map. However, to represent the geometry of the model, an additional *embedding* structure is used. If only linear geometries are required, only the 0-dimensional point embeddings are actually needed. These store the coordinates of each vertex.

Since an α_i involution connects adjacent i-cells in the primal graph of a d-dimensional model, per definition α_{d-i} does so in the dual graph. These can therefore be easily swapped to convert a graph into its dual. For the 4D case, α_4 connects corresponding involutions for the dual of the nodes (0-cells), α_3 for the edges (1-cells), α_2 for the faces (2-cells), α_1 for the volumes (3-cells), and α_0 for the 4-cells.

Unlike the DHE that permits us to represent both the primal and the dual graph simultaneously, G-maps permits us to directly represent either one or the other. Transforming a 4D cell complex into its dual is however a straightforward operation, the combinatorial part of it being performed
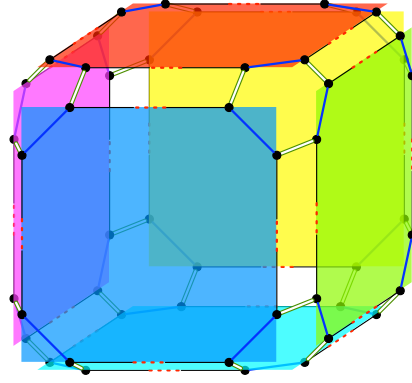
(a) A G-map representation of a cube.
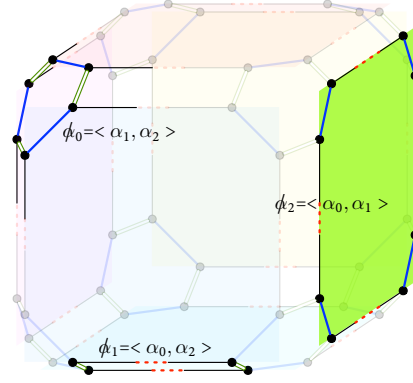
(b) The α_i operator obtains all the darts belonging to a specific i-cell. Thus, α_0 obtains the darts belonging to a vertex, α_1 those belonging to an edge, and α_2 those belonging to a face.
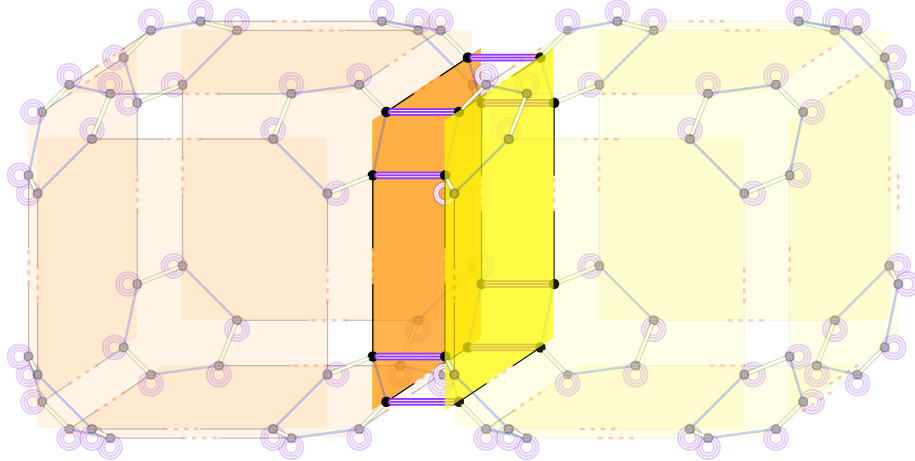
(c) A G-map representation of two cubes. Note how the individual cubes have identical involutions to those of (a), with the addition of an α_3 involution that connects the two cubes at their common face. In the other darts, this involution is not used.

Figure 6: A 3D G-map representation of a pair of adjacent cubes, showing the α_0 (dashed), α_1 (solid), α_2 (double), α_3 (triple), and α_i operators.

in linear time. Alternatively, any cell's dual can be directly obtained from the graph by interpreting an α_i involution as an α_{d-i} one. This is similar to how a Voronoi diagram is instead often manipulated from a Delaunay triangulation, cf. Boots (1974). Duality in the combinatorial structure of G-maps is therefore trivial to obtain, and can be done in real time.

Meanwhile, a geometric interpretation of the dual graph is also simple to get. Assuming linear geometries, only the point embeddings for the dual of the 4-cells need to be generated, e.g. using the centroid of the 4-cell, or simply an average of the point embeddings of the 0-cells in the boundary of the 4-cell. These are both easy to obtain using the $< \not\alpha_4 >$ orbit corresponding to the 4-cell. Note that if the initial model (primal graph) is bounded, the dual representation will have unbounded cells. The running time of the geometric part of the duality transformation depends on the manner in which new point embeddings are computed. When each of these can be computed in constant time (e.g. using a few darts in their orbit), the entire transformation can be done in linear time. Otherwise, the complexity will be higher.

The process of the duality transformation of a 4D G-map is shown in the algorithm below. For simplicity of explanation, three things are assumed to exist: an additional pointer to store the new point embeddings, a global list of embeddings, and a pointer from each embedding to a dart in its boundary. Note however that these are not strictly necessary. Their existence depends on the manner in which G-maps are implemented.

Algorithm 1: DUALTRANSFORM(G)

Input : A 4D G-map G
Output: The dual transformation of G
foreach *4-Cell c in G* **do**
    $p \leftarrow$ the centroid of c
    $O \leftarrow\ <\not\alpha_4>(c.dart)$
    foreach *Dart d in O* **do**
        $d.embedding_{4D} \leftarrow p$
    end
end
foreach *Dart d in G* **do**
    swap($d.\alpha_0$, $d.\alpha_4$)
    swap($d.\alpha_1$, $d.\alpha_2$)
    swap($d.embedding_{0D}$, $d.embedding_{4D}$)
end

For consistency in our dual representation, we assume that there is an external (unbounded) cell. This ensures that applying the dual operation twice returns a model that is topologically equivalent to its original representation. The transformation of a 2D G-map into its dual is shown step by step in Figure 7.

(a) A 2D G-map representation of 3 adjacent triangles.

(b) α_0 and α_2 are swapped, and a point embedding for each triangle is created.

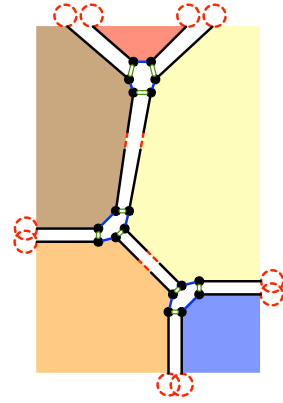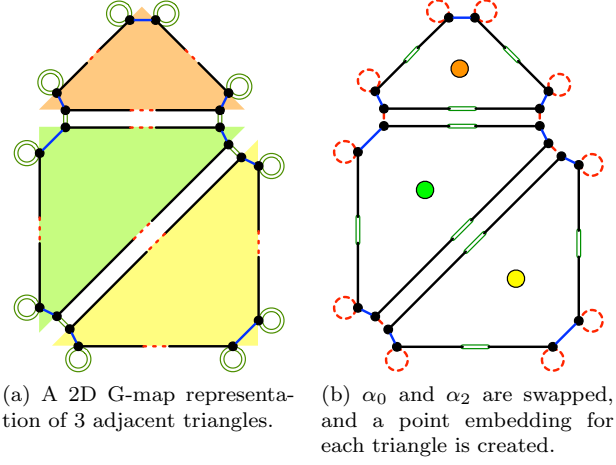(c) The final result after linking to the new point embeddings.

Figure 7: A step by step transformation of a 2D G-map into its dual representation.
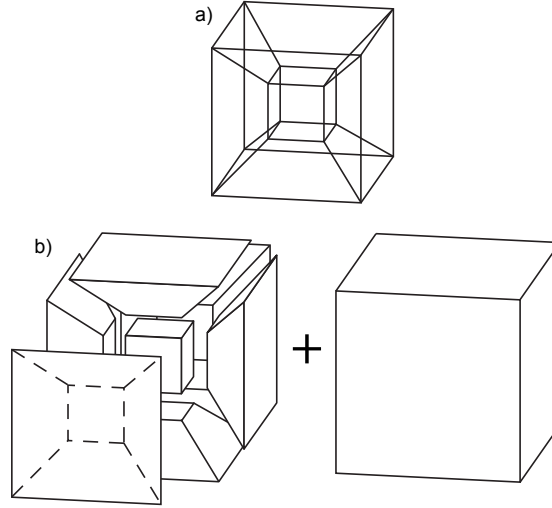
Figure 8: A 3D projection of a tesseract: a) a tesseract is a simple polychoron; b) a tesseract is bounded by eight cubes, and can be represented as a cellular complex of these cubes. Note that they have different shapes due to the projection used.

4   Storage of a 4D cell complex and its dual

In this section we consider a 4D cell complex consisting of adjacent tesseracts and its dual. A tesseract, also known as a 4-cube or cubic prism, is the four dimensional analogue of a cube. As shown in Figure 8, it is a closed four-dimensional polytope bounded by eight cubes. It contains 8 cubical 3-cells, 24 square 2-cells, 32 1-cells, and 16 0-cells.

4.1   Dual half-edge

Using the 3D DHE, a single tesseract can be represented as a complex of seven internal and one external 3-cells. This perfectly fits to the DHE concept of models enclosed by the external cell. Once each tesseract is created individually, it is necessary to link them into a 4D complex. Using the DHE, at this moment only adjacency by a shared 3-cell (a cube) is taken into consideration—other relationships between lower dimensional cells (i.e. by a shared face, edge, and vertex) are not allowed. This significantly simplifies the problem.

To extend the concepts of the original 3D DHE to 4D, it is necessary to introduce the concept of an external 4-cell into the model. In a model with only one object, e.g. a single tesseract, there will be two 4-cells—internal and external—connected into a complex by adjacent 3-cells. Thus, in the dual of this model, there would be two vertices, which correspond to the internal and external 4-cells. The vertex corresponding to the internal

one can be calculated as the centroid of the tesseract; while the node for the external one may be located at infinity—these two dual nodes are connected by eight dual edges representing all the bounding 3-cells of the internal and external 4-cells. Technically, each 3-cell is represented by a bundle of dual edges, but since the bounding nodes of the edges in the bundle are geometrically the same, and the edges are connected in a radial cycle, they are considered as a single edge. Each cube in this example is represented by a bundle of 12 dual edges. Since every primal half-edge is associated with a dual half-edge, the number of dual edges is the same as the number of primal edges of the cell.

A simple calculation determines that the number of atomic elements, dual half-edges, required to represent the above model is 384: there are two tesseracts (internal and external) consisting of eight cubes each; each cube consisting of 12 edges; and each edge consisting of two DHEs. It should be noted that the 3D dual graph originally used to connect 3-cells of a complex is replaced by the new 4D graph connecting 4-cells. However, at the current stage of our research we cannot determine all the DHE connections between the dual edges, and the exact number of pointers necessary to represent these connections.

4.2   Generalised maps

A complex of two or more adjacent tesseracts can be represented using a 4D generalised map (4-G-map), in which each dart has 5 involutions (α_0 to α_4). The construction of the model proceeds incrementally, starting from the vertex level. A vertex is defined by a point embedding, where its 4D coordinates are stored. An edge is defined by creating a pair of darts, linked to each other along the α_0 involution. Four of these edges, linked along their shared vertices at the α_1 involution, form a square face. Six of these faces, linked along their shared edges at the α_2 involution, form a cube. Eight of these cubes, linked along their shared faces at the α_3 involution, form a tesseract. The resulting tesseract is thus formed of $2 \cdot 4 \cdot 6 \cdot 8 = 384$ darts and 16 point embeddings. Since each dart requires 6 pointers (one for each involution plus one for its point embedding), there are 2304 pointers in the combinatorial structure.

The α_4 involutions have not been used up to this point. These are however used to link 3D-adjacent tesseracts together. Since no additional pointers are required, the total storage used for a cell complex of 4D tesseracts is the sum of the storage for each individual tesseract.

To obtain the dual of this model, the procedure described in Section 3.2 may be used. The α_i and α_{d-i} involutions are first swapped in the combinatorial structure. In this manner, α_0 becomes α_4 and vice versa, and α_1 becomes α_3 and vice versa. Afterwards, a new point embedding at the centre of each tesseract is created and linked to the darts on its boundary.

5   Discussion

We have shown how it is possible to store the dual graph of a 4D object by applying and extending existing data structures. G-maps already of-

fer this possibility, although simultaneous storage of both graphs is not possible. The dual half-edge offers this possibility and is thus a promising alternative, especially as the dual graph is updated automatically while the primal is modified.

We also envision being able to use the 4D dual graph for various applications, navigation in 4D being an interesting possibility. For instance, it would make it possible to create a 3D indoor and outdoor way-finding application, where a user can select any given start and end points, and be given the best 3D route at any point in time, taking into account topological changes (e.g. a connecting corridor being only open during office hours).

We also plan to work on duality when holes/cavities are allowed in any dimension, up to 4D. An example of a 4D hole could be a section of a building being closed due to refurbishing work and thus inaccessible and removed from the graph. Note however that this assumption might not be true for all applications, e.g. emergency response In a 3D representation, there would not be a natural connection between the building before and after the construction work, but it would be there in the 4D dual graph. This will allow us to fully utilise existing spatial datasets, and at the same time be able to represent a greater variety of situations.

6   Acknowledgment

References

Bruce G. Baumgart. A polyhedron representation for computer vision. In *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*, pages 589–596, 1975.

H. Bieri and W. Nef. Elementary set operations with d-dimensional polyhedra. In *Computational Geometry and its Applications*, volume 333 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin / Heidelberg, 1988.

Pawel Boguslawski. *Modelling and Analysing 3D Building Interiors with the Dual Half-Edge Data Structure*. PhD thesis, University of Glamorgan, 2011.

Pawel Boguslawski, Christopher M. Gold, and Hugo Ledoux. Modelling and analysing 3D buildings with a primal/dual data structure. *ISPRS Journal of Photogrammetry & Remote Sensing*, 66:188–197, 2011.

Barry Boots. Delaunay triangles, an alternative approach to point pattern analysis. In *Proceedings, Association of American Geographers, 6*, pages 26–29, 1974.

C. Braid, I., C. Hillyard, l., and A. Stroud, I. Stepwise construction of polyhedra in geometric modelling. In K. W. Brodlie, editor, *Mathematical Methods in Computer Graphics and Design*, pages 123–141. Academic Press, 1980.

Erik Brisson. Representing geometric structures in d dimensions: Topology and order. In *Proceedings 5th Annual Symposium on Computational Geometry*, pages 218–227, Saarbrücken, West Germany, 1989. ACM Press.

Rizwan Bulbul, Farid Karimipour, and Andrew U. Frank. A simplex based dimension independent approach for convex decomposition of nonconvex polytopes. In *Proceedings of Geocomputation 2009*, 2009.

Maciej Dakowicz and Christopher M. Gold. Extracting meaningful slopes from terrain contours. *International Journal of Computational Geometry and Applications*, 13(4):339–357, 2003.

J. Edmonds. A combinatorial representation of polyhedral surfaces. *Notices of the American Mathematical Society*, 7, 1960.

Christopher M. Gold. Problems with handling spatial data—the Voronoi approach. *CISM Journal*, 45(1):65–80, 1991.

Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

Hugo Ledoux and Christopher M. Gold. Modelling three-dimensional geoscientific fields with the Voronoi diagram and its dual. *International Journal of Geographical Information Science*, 22(5):547–574, 2008.

Ickjai Lee and Mark Gahegan. Interactive analysis using Voronoi diagrams: Algorithms to support dynamic update from a generic triangle-based data structure. *Transactions in GIS*, 6(2):89–114, 2002.

J Lee and M.-P. Kwan. A combinatorial data model for representing topological relations among 3D geographical features in micro-spatial environments. *International Journal of Geographical Information Science*, 19(10):1039–1056, 2005.

J Lee and Sisi Zlatanova. A 3D data model and topological analyses for emergency response in urban areas. In S. Zlatanova and Jonathan Li, editors, *Geospatial Information Technology for Emergency Response*, pages 143–168. Taylor & Francis, 2008.

Bruno Lévy and Jean-Laurent Mallet. Cellular modeling in arbitrary dimension using generalized maps. Technical report, ISA-GOCAD, 1999.

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

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

Yuanxin Liu and Jack Snoeyink. The "far away point" for Delaunay diagram computation in $\mathbb{E}^d$. In *Proceedings 2nd International Symposium on Voronoi Diagrams in Science and Engineering*, pages 236–243, Seoul, Korea, 2005.

Martti Mäntylä. *An introduction to solid modeling*. Computer Science Press, New York, USA, 1988.

H. Masuda. Topological operators and boolean operations for complex-based nonmanifold geometric models. *Computer-Aided Design*, 25(2): 119–129, 1993.

A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.

Donna J. Peuquet. *Representations of Space and Time*. Guilford Press, 2002.

Jonathan Raper. *Multidimensional geographic information science*. Taylor & Francis, London, 2000.

Jonathan Richard Shewchuk. Sweep algorithms for constructing higher-dimensional constrained delaunay triangulations. In *Proceedings of the 16th Annual Symposium on Computational Geometry, Hong Kong*, pages 350–359, 2000.

Cathy Sohanpanah. Extension of a boundary representation technique for the description of n dimensional polytopes. *Computers & Graphics*, 13 (1):17–23, 1989.

Peter van Oosterom and Martijn Meijers. Towards a true vario-scale structure supporting smooth-zoom. In *Proceedings of the 14th ICA/ISPRS Workshop on Generalisation and Multiple Representation, Paris*, 2011.

Peter van Oosterom and Jantien Stoter. *5D Data Modelling: Full Integration of 2D/3D Space, Time and Scale Dimensions*, chapter Proceedings 6th International Conference GIScience 2010, pages 311–324. Springer, 2010.

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