Simultaneous Storage of Primal and Dual Three-Dimensional Subdivisions

Hugo Ledoux[*] Christopher M. Gold

GIS Research Centre, School of Computing, University of Glamorgan
Pontypridd CF37 1DL, Wales, UK.
tel: +44 (0)1443 483211 fax: +44 (0)1443 482715
`hledoux@glam.ac.uk` `christophergold@voronoi.com`

Abstract

We propose a new general-purpose data structure useful for a variety of three-dimensional applications. The data structure has the characteristic of storing simultaneously the primal and dual subdivisions of a three-dimensional manifold. We argue in this paper that storing both subdivisions, for instance the Voronoi diagram and the Delaunay tetrahedralization, can be beneficial for many application domains, notably for the modelling of datasets in geosciences or for representing boundaries of real-world features. Our structure is an extension of the well-known *quad-edge* data structure used for representing two-dimensional manifolds. We describe the basic properties of this *augmented quad-edge* structure, along with the navigation operators, and we also demonstrate its usefulness with some examples of applications.

Keywords: Data structure, Three-dimensional modelling, Duality, Voronoi diagram, Delaunay tetrahedralization.

1 Introduction

The topological data structures commonly found in geographical information systems (GIS) or other modelling systems have been used—with success—for many years to model two-dimensional (2D) objects and surfaces (e.g. a digital terrain model where the elevation is treated as an attribute). The TIGER (Boudriault, 1987), DCEL (Muller & Preparata, 1978) and ARC/INFO (Morehouse, 1985) formats are all examples of such structures. Although many application domains require more than that, data structures to model three-dimensional (3D) solid objects (not only their surface, but also their interior) are still not well-studied and can even be considered to be in their infancy.

[*]Corresponding author.

Examples of domains where 3D data structures are required include computer graphics/games, CAD systems, medical imaging, and the modelling of cities or geoscientific datasets.

In this paper, we propose a new general-purpose data structure useful for a variety of applications where 3D solid objects are involved. To represent such objects in computers, the space they cover has to be decomposed, or subdivided, into finite parts, i.e. into volume elements that we refer to as *cells*. For the purpose of this paper, we are particularly interested in data structures that permit us to store not only one subdivision of a 3D object, but simultaneously both that subdivision (which is refer to as the *primal*) and another related subdivision called the *dual*. The terms 'primal' and 'dual' refer to the concept of *duality* in mathematics, as explained in Section 3. In brief, it means that two subdivisions are inter-connected: they represent the same thing, just from a different point of view. The most known and used example of primal and dual subdivisions is the duality that exists in any dimensions between the Voronoi diagram (VD) and the Delaunay triangulation (DT).

Our claim in this paper is that storing both the primal and the dual subdivisions can be beneficial for a broad range of applications, and that doing so offers a more powerful and flexible solution than the other known 3D data structures (see Section 2 for a review). As explained in Section 5, keeping both subdivisions can optimise certain spatial analysis operations in GIS, and permit new applications involving the volume and the boundary of an object. Also, it offers a flexible solution, because attributes for any elements of both subdivisions can be stored, which is of considerable importance for the modelling of real-world features. On the other hand, keeping two subdivisions at the same time obviously has drawbacks: it increases the requirements for storage, and the construction/modification algorithms will be slower. We nevertheless believe that in many real-world applications the major constraint is not the speed of construction of the topological models of a large number of points or the space needed in memory, but rather the ability to interactively construct, edit (by deleting or moving certain points) and query (interpolation, extraction of implicit surfaces, etc.) the desired model. See Anselin (1999) and Gold (1993) for discussions about the advantages of interactive modelling.

We present in Section 4 a new data structure to store and manipulate simultaneously the primal and dual subdivisions of a 3D object. The structure, called the *augmented quad-edge* (AQE), is based on the *quad-edge* structure (Guibas & Stolfi, 1985) that has been used for many years to represent the primal and dual subdivisions of 2D objects. The basic properties, the construction and navigation operators, and the storage costs of the AQE are also presented in Section 4. We believe the primary feature of the data structure to be ease of use in managing and navigating 3D objects. While it needs somewhat more storage than other structures, it is computationally efficient, requiring no searching operations in order to move from cell to cell.

2 Related Work

Before discussing the alternative data structures, we need to introduce some concepts and definitions. When dealing with d-dimensional objects, an important concept is that of a *manifold*. Informally, a d-manifold is an object for which the neighbourhood of every one of its points resembles a d-dimensional space. A 2-manifold is therefore a surface which, it should be pointed out, can be embedded in 3D space (see Figure 1a); an obvious example is the surface of the Earth, for which near to every point the environment is equivalent to a plane. An example of a 3-manifold is the entire Earth (its interior) because the neighbourhood of every point is equivalent to a 3D region (see Figure 1b). Closely related to the concept of a subdivision of a 3-manifold is that of a 3D cell complex. A cell complex is formed by a collection of k-cells (where $0 < k \leq d$, such that a 0-cell is a vertex, a 1-cell an edge, a 2-cell a face and a 3-cell a polyhedron) such that no k-cell punctures the interior of any other k-cell. In the following, unless explicitly stated, a cell is always assumed to be a 3-cell. Note that a 3D cell complex is the same as the tessellation of 3D space into non-overlapping polyhedra.

Data structures to represent 3D objects have been developed in many different fields, and we would classify the most common 3D spatial data structures as follows. Note that this is not a complete review, our purpose is to introduce some of the methods used in other disciplines.

Constructive solid geometry (CSG): Solid objects are represented as Boolean combinations (union, intersection and difference) of simpler objects, which are typically of simple shapes such as cylinders, cones, spheres or pyramids (see Requicha (1982) for more details). CSG is mostly used in CAD systems to represent complex models of man-made features (e.g. aircrafts or engines).

Boundary representation (*b-rep*): One 3D object—it must be a 2-manifold—is represented by its boundary surface, using primitive objects (vertices, edges and faces). Boundary modelling is more flexible than CSG because irregular objects can be modelled, but it requires more work as the information about the connectivity of the faces needs to be valid at all times. One example of the use of such a structure is the modelling of cities in a GIS where each building is represented individually by a *b-rep* (see Zlatanova et al. (2004) for a review of the topological models used).

Regular decomposition: The *space* covered by an object is arbitrarily decomposed into cells of regular shape (usually cubes in 3D). The term 'voxel' is used as the 3D equivalent of pixel. This structure, or closely related hierarchical decompositions (Samet, 1984), have been used in many application domains to represent continuous phenomena.

Irregular decomposition: The *object* is decomposed into cells of different size and/or shape. The cells can all have the same shape (e.g. tetrahedra), or be arbitrary polyhedra. It is equivalent to a subdivision or a tessellation of a 3-manifold.

Non-manifold structures: These are used to represent non-manifold objects, i.e. objects for which the neighbourhood of a point does not resemble a 3D region (see Figure 1c). See De Floriani & Hui (2003) for one implementation of such a structure.

Let us now describe in more detail some of the most relevant data structures. To represent 2-manifolds, only the edges and vertices of a subdivision are usually stored, and for each edge the connected edges are stored; this permits us to store implicitly the faces of the boundary. The most popular structures are the *half-edge* (Mäntylä, 1988), the DCEL (Muller & Preparata, 1978) and the *winged-edge* (Baumgart, 1975). Guibas & Stolfi (1985) also propose the *quad-edge*, which stores simultaneously any primal and dual subdivisions of a 2-manifold. The structure, which is formally described in Section 4.1, comes with an algebra for the construction and the navigation of the subdivisions.

Many data structures to store arbitrary 3D subdivisions are extensions of work done in 2D. Examples are the *half-face* of Lopes & Tavares (1997) (extension of the half-edge) and the *Generalized Maps* (G-Maps) (Bertrand et al., 1993; Lienhardt, 1994). The latter structure, which is actually valid to represent a broader class of objects called *cellular quasi-manifolds*, can be seen as a generalization to d-manifolds of boundary models as each k-cell (where $0 < k \leq d$) is recursively decomposed into cells of lower dimensionality, and the topological relationships between adjacent k-cells are kept. The resulting structure is thus very space-consuming, as each lower-dimensionality cell must be stored in cells of higher dimensionality: a vertex shared by five edges is for instance stored in each of the five edges. It is nevertheless being used in a commercial system for the modelling of geoscientific data[1]. Although G-Maps store only one subdivision, it offers efficient functions to extract all the elements of the dual subdivision. One of the few structures permitting the simultaneous preservation of primal and dual subdivisions of 3-manifolds is the *facet-edge* structure of Dobkin & Laszlo (1989). As its name implies, the atom is a pair formed of a face and an edge, and the next facet-edge pairs (one in each direction) around the face and the edge (incident faces to the edge) are stored. Unlike G-Maps, a face shared by two cells needs only be stored once. The facet-edge structure comes with a set of operations to modify cells and to navigate within both subdivisions. Its generality makes navigation within a single cell impossible, and hence the authors suggest storing extra information for each edge and using the quad-edge operators. Also, unlike the quad-edge that is being used in many implementations of the 2D VD/DT, the facet-edge has been found difficult to implement in practice and, to our knowledge, has not been used in 'real projects'.

As Aurenhammer (1991) demonstrates, the VD and the DT are fundamental concepts in computer science because they are useful in a wide variety of appli-

[1]The gOcad system: `www.gocad.org`

cations, and once they are constructed they permit us to solve many geometric problems. In fact, they are such important concepts that it is worth mentioning data structures to store these in particular. In practice, most of the algorithms and implementations available to construct the 3D VD/DT store only the DT and perform their topological operations on tetrahedra, and if needed the VD is extracted afterwards. This is usually justified by the fact that the knowledge of either structure implies the knowledge of the other, and also because managing and storing simplices—having a fixed number of vertices and adjacent cells—over arbitrary polyhedra simplifies the topological operations and permits the design of simpler data structures. The resulting implementation will therefore be more efficient, both in terms of speed and memory. The simplest data structure considers the tetrahedron as being its atom and stores each tetrahedron with four pointers to its vertices and four pointers to its adjacent tetrahedra. This is a very space-efficient structure that yields fast implementations, and it is therefore the structure of choice in many projects. The `CGAL` library notably uses it, but has added to each vertex a pointer to one of its incident tetrahedra to speed up the extraction of the Voronoi cells (Boissonnat et al., 2002). It is also possible to store the tetrahedra implicitly by considering a data structure where the atom is a triangular face having three pointers to its vertices and six pointers to its adjacent faces, as described by Shewchuk (1997). However, storing only the DT has major drawbacks if one wants to work with the VD. It is for example difficult to assign attributes to Voronoi vertices or faces, and moreover the computation of the VD is an expensive operation, as explained in Section 5.2.

3 Duality in Three-dimensional Space

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 graph of a given graph. Let G be a planar graph, as illustrated in Figure 2a. Note that G can both be seen as a subdivision of a 2-manifold, or as a 2D cell complex. The dual graph $G^\star$ has a vertex for each polygon in G, and the vertices in $G^\star$ are linked by an edge if and only if the two corresponding dual polygons in G are adjacent (in Figure 2b, $G^\star$ is represented with dashed lines). Notice also that each polygon in $G^\star$ corresponds to a vertex in G. Consider now the polygonal map in Figure 2c, where each polygon (a map object) has certain attributes, and where the boundaries between two map objects also have some sort of attributes, e.g. the boundary form or the flow. Notice that the boundaries do not characterise *per se* any of the objects, but rather the adjacency relationships that exist between any two objects (Gold, 1991). It is therefore natural to store the relationships between two given map objects in the edge that is dual to their shared boundary (Figure 2d).

The best known example of a graph and its dual is the Voronoi diagram and the Delaunay triangulation. In this case, geometric constraints are added so that the graphs respect some criteria. Let S be a set of points in the Euclidean space $\mathbb{R}^2$. Many triangulations (a triangulation is a graph) of S are possible, but only one has the property that the interior of the *circumcircle* of each triangle is empty of any other point in S: the Delaunay triangulation, as shown in Figure 3. The dual graph of the DT is the VD: each triangle in S becomes a vertex in the dual (it is located at the centre of the circumcircle of the triangle) and two vertices in the VD are linked by an edge if and only if their respective dual triangles are adjacent. The resulting polygon dual to a point $p \in S$ is a Voronoi cell, denoted $\mathcal{V}_p$, which is defined by the set of points $x \in \mathbb{R}^2$ that are closer to p than to any other point in S. From a mathematical point of view:

$$\mathcal{V}_p = \{x \in \mathbb{R}^2 \mid \|x - p\| \leq \|x - q\|, \; \forall q \in S\}.$$

The union of the Voronoi cells of all points p in S form VD(S). In any dimensions, a Voronoi cell around a point p is always convex, and its size depends on the configuration of neighbouring points. The VD and DT are very important structures in a wide range of domains, see Aurenhammer (1991) for an excellent survey.

Both the VD and the DT generalise to any dimensions. The Voronoi cell $\mathcal{V}_p$ of a point p in $\mathbb{R}^3$ has the same definition but becomes a polyhedron. The generalisation to 3D of the Delaunay triangulation is the Delaunay tetrahedralization: each triangle becomes a tetrahedron that satisfies the *empty circumsphere* rule. The duality between the two structures in 3D is also similar: each tetrahedron of the DT becomes a vertex (located at the centre of the circumsphere of the tetrahedron), and two vertices in the VD are linked by an edge if and only if their two respective dual tetrahedra are adjacent.

More formally, the mapping between the elements of primal and dual subdivisions in $\mathbb{R}^d$ is as follows. Let C be a k-cell of any structure, the dual cell of C in $\mathbb{R}^d$ is denoted by $C^\star$ and is a $(d - k)$-cell. Therefore, in 3D, a point becomes a polyhedron, and vice versa; and an edge becomes a face, and vice versa. For example, a Delaunay vertex p becomes a Voronoi cell (Figure 4a), a Delaunay edge α becomes a Voronoi face (Figure 4b), a Delaunay triangular face κ becomes a Voronoi edge (Figure 4c), and a Delaunay tetrahedron τ becomes a Voronoi vertex (Figure 4d).

4 Augmented Quad-Edge

4.1 Quad-Edge Structure

The quad-edge data structure (Guibas & Stolfi, 1985) was developed to represent simultaneously the primal and dual subdivisions of a 2-manifold, and also to navigate from edge to edge in these subdivisions. Each quad-edge represents one geometrical edge of a subdivision, and this edge is decomposed into four directed edges: two in the primal, and their two respective dual edges. Each directed edge is represented by a *quad*, which, in a given subdivision, will either point to one vertex or to a face; in the dual, a quad points to the dual of what it is pointing to in the primal. Its advantages are firstly that there is no distinction between the primal and the dual representation (it is symmetric with respect to edges and faces/polygons), and secondly that all operations are performed as pointer operations only, thus giving an algebraic representation to its operations. Figure 5a shows the basic structure, with four branches (quads) for each edge of the graph being stored (the dual edges are not drawn), one for each of the incident vertices or faces. There are three pointers on each quad: one to the vertex or face object (*org*), one to the next anticlockwise quad-edge around that object (*next*), and one to link the four quads together in a loop (*rot*). Thus all vertices or faces have complete topological loops around them. Notice that the *rot* operation actually permits us to navigate between the primal and the dual subdivisions.

Only two commands are used to modify a graph: *MakeEdge* to create a new edge on a manifold, and *Splice* to connect/disconnect quad-edges together (illustrated in Figure 5b). In the simplest case, *Splice* connects two separate *next* loops, joining the two nodes together, and at the same time splitting the *next* loop around the common face. *Splice* is its own inverse.

4.2 One Dimension Higher

The augmented quad-edge (AQE) uses the 'normal' quad-edge, which is valid for any 2-manifolds, to represent each 3-cell of a 3D complex, in either space. For instance, each tetrahedron and each Voronoi cell are independently represented with the quad-edge, which is akin to a boundary representation (*b-rep*). With this simple structure, it is possible to navigate within a single cell with the quad-edge operators, but in order to do the same for a 3D cell complex two things are missing: a way to 'link' adjacent cells in a given space, and also a mechanism to navigate to the dual space. First, notice that in this case two of the four *org* pointers of a quad-edge point to vertices forming the 2-manifold, but the other two (which in 2D point to the dual, or a face) are not used in 3D. Remember also that in 3D the dual of a face is an edge. Our idea is therefore to use this dual edge to 'link' two cells sharing a face: the unused face pointers simply point to their dual edge. This permits us to 'link' cells together in either space, and also to navigate from a space to its dual. Indeed, we may move from any quad-edge with a face pointer to a quad-edge in the dual cell complex, and from there we may return to a different 2-manifold in the original cell complex if needed.

A quad-edge is divided into four quads q, and there exist two types of quads: a q_f points to a face, and a q_v points to a vertex. One *rot* operation applied to a q_f returns a q_v, and vice-versa. A q_f identifies uniquely, like the facet-edge (Dobkin & Laszlo, 1989), a pair (face, edge). Therefore q_f has also a linked quad $q_f^\star$ in the dual that is defined by (face*, edge*).

One issue remains to be resolved: as each face is penetrated by several dual edges, a consistent rule must be defined to select the appropriate one. Indeed, with the AQE, the dual edge to a face has to be stored for all the dual cells sharing that edge. A triangular face has for example three dual edges since each of its three vertices becomes a cell in the dual. A q_v has its *org* pointer set to a node, and a q_f has its *org* pointer set to $q_f^\star$. The pointer to $q_f^\star$ from q_f is called *through*, as shown in Figure 6. The quad $q_f^\star$ is defined as belonging to the dual cell which encloses the node pointed to by $q_f.rot = q_v$. This is sufficient to define the *through* pointer structure. In Figure 6, the triangular face *abc* has three dual edges and may be represented by the quad q_f (the black quad) on the edge *ab*. The dual edge of *abc* is *uv*, and the grey face represents the face that is dual to the edge *ab*. The operation $q_f.through$ gives $q_f^\star$ which is situated on the edge *uv* and on the face dual to *ab*; this face belongs to the cell dual to vertex *b* since $q_f.rot.org$ gives *b*. Note also that $q_f.next.through$ gives a quad on a face which belongs to the cell dual to the vertex *a*; and $q_f.next.next.through$ to *c*.

4.3 Navigation and Construction Operators

As mentioned earlier, because the quad-edge structure is used to represent each cell of a 3D cell complex, the 'usual' navigation operators (*next* and *rot*) described in Section 4.1 can be used to navigate within a 3-cell. Only one new operator is needed to navigate within a 3D cell complex and to be able to visit every polyhedron, face, node or quad-edge: the *through* operator, as described in the previous section.

Let q_f be a quad and $q_f^\star$ its dual quad, then the following properties are valid:

1. $q_f^\star = q_f.through$
2. $q_f = q_f.through.through = q_f^\star.through$

Different higher-level navigation operators can also be defined. One that is particularly useful is an operator for 'jumping' from a cell to another adjacent cell, in either space. We call this navigation operator *adjacent*. Let q_f be a quad on the boundary of a common face between two cells, as shown in Figure 7. We want to 'jump' to the quad r_f representing the same pair (face, edge) as q_f but on the adjacent cell. We do this by using the dual structure to navigate. Starting from q_f, $q_f.through$ gives $q_f^\star$, and $q_f^\star.next$ gives a quad whose *through* pointer gives a quad in the original space but in a neighbouring cell. The *rot* operator is used twice to set the result to r_f. Thus,

$$q_f.adjacent = q_f.through.next.through.rot^2$$

Other relations are also interesting. First, the *adjacent* operator is symmetrical: $q_f = q_f.adjacent.adjacent$. Second, if q_f and r_f are adjacent quads as in Figure 7 (such that $r_f = q_f.adjacent$), notice that their respective dual quads $q_f^\star$ and $r_f^\star$ are also adjacent, i.e. they represent the same pair (face, edge) but are part of different 3-cells.

New construction operators are required to build a 3D cell complex and its dual. The creation of new independent cells, e.g. a Delaunay tetrahedron or a Voronoi cell, can be done with only the construction operators *MakeEdge* and *Splice* of the quad-edge structure because each cell is represented as a

2-manifold. The other construction operator needed is for the assemblage or linkage of two cells. Again, this is done fairly easily as only the *through* pointers of all q_f belonging to a common face need to be set (reset) to link (un-link) two cells.

4.4 Properties

We have attempted to develop a data structure with a variety of properties. The AQE integrates the *b-rep* representations and the irregular and regular decompositions under the same structure, as described in Section 2. Its flexibility, and the fact that the dual of a 3D cell complex is also stored, makes the AQE a valuable choice for many applications, as we discuss in Section 5. It should be noticed that the AQE is only valid for cell complexes, and that subdivisions where 'dangling' elements are present cannot be represented with the structure in its current form.

The proposed approach preserves the lower dimensionality (2D) navigation and construction operations, as well as keeping the underlying quad-edge properties in 3D. As a consequence, the data structure is directly navigable and forms an 'edge algebra', allowing us to combine our pointer operations by following well defined rules, and to show the equivalence of different ways of getting to the same destination. Also, every operation has an inverse. To permit the navigation between 2-manifolds, the dual has to be explicitly stored within the structure. Each volumetric element uses the quad-edge (with 2D primal and dual edge elements) for navigation around its boundary, and the same is thus required for the dual structure, as we wish the same set of operations to be valid in each space.

Furthermore, the AQE has the major benefit of being *locally* modifiable, i.e. if new elements need to be inserted (deleted) in (from) a model a complete reconstruction of the model is not required. We have already implemented with the AQE an algorithm to construct and store both the 3D DT and the VD. It is based on local transformations, called *flips*, to modify a tetrahedralization (Joe, 1991). A DT is constructed incrementally by inserting one at a time the points in a set and updating the DT—with flips—after each insertion. Our modifications involve the update of the dual structure when a flip is performed; because the flips are simple topological operations involving a fixed set of tetrahedra, the operations to update the VD are relatively simple and can be described using algebraic statements exclusively. This simplicity also means that vertices can be deleted from a 3D VD/DT (Ledoux et al., 2005), which is of great value for many applications. More details concerning the construction and the modification of the 3D VD/DT with the AQE will be published in another paper.

4.5 Storage Costs and Comparisons

We analyse in this section the storage costs of the AQE when it is used to represent both the DT and the VD.

With the AQE, each cell (tetrahedron or Voronoi cell) is stored separately. Each tetrahedron contains: six edges, each represented by four quads containing three pointers (*org*, *next* and *rot*). This makes a total of 72 pointers. Notice that every q_f has one and only one $q_f^\star$, and also that the same is true for every q_v (there is always one q_v when a *rot* is applied to a given q_f). As a result, the total number of pointers for the dual is also 72, which makes a total of 144 pointers for each tetrahedron.

This is obviously more than the data structures used to store only the DT. For example, the tetrahedron-based structure described in Section 2 uses only eight pointers per tetrahedron. To our knowledge, the only data structure preserving both a primal and dual 3D subdivision is the facet-edge (Dobkin & Laszlo, 1989). With it, each pair (face, edge) contains (considering both the primal and the dual): four pointers to vertices (two Delaunay and two Voronoi), and also four pointers to the *next* facet-edge (the navigation requires the use of the dual to access the adjacent facet-edges around the face and the edge). There are three pairs per triangular face, and a tetrahedron contains four faces. However, with this structure, the common faces between two tetrahedra are not stored twice—and we can state that each face is shared by two tetrahedra, except the ones on the convex hull of the data set. As a result, the total number of pointers per tetrahedron is 48, but, as explained previously, extra information needs to be stored if one wants to use the quad-edge operators to navigate within a single cell.

To summarise, the AQE, as described in Section 4, is about three times more space-consuming than the facet-edge, in the worst case. A factor of two is consistent with our initial objective to maintain the primal and dual structures simultaneously, and to use the same set of operations in each space. This can be achieved by compressing the quad-edge as Guibas and Stolfi proposed: the *rot* pointers between quads are not explicitly stored and the last two bits of each pointer are used to identify each quad of a quad-edge object.

5 Applications

Many applications in 3D are based on the idea of subdividing or tessellating the space to be able to model objects. This section gives a few examples of applications for which the dual of a subdivision is meaningful, and therefore is worth storing. Doing this can optimise some of the operations necessary for an application, solve a problem that data structures storing only one subdivision have, or offer a more flexible option for the storage of attributes.

5.1 Crust and Skeleton

In recent years, tetrahedralizations have been used more and more to model the interior and the boundary of objects. The principal application is in engineering where the subdivision of an object is used in the finite element method or other numerical methods for solving partial differential equations. Delaunay tetrahedra are usually preferred because of their properties, but are sometimes *refined* so that they respect some criteria (see Shewchuk (1997) for a discussion). This is mostly the reason why the efforts of the computational geometry community have been directed towards data structures to efficiently store tetrahedra, and only tetrahedra.

But some applications are only possible when both the DT and the VD are known *explicitly*, i.e. that properties of the DT and the VD are both used to solve a problem. Obvious two-dimensional examples are the definitions of the crust and skeleton of a sufficiently well-sampled polygon. It has been shown that these two structures are subsets of respectively the DT and the VD (Amenta et al., 1998), and the extraction of either the crust or the skeleton requires the knowledge of both the DT and the VD. Gold & Snoeyink (2001) show that the crust/skeleton concept helps in many cartographic problems such as the reconstruction of terrain models from contour lines, the estimation of a river network from boundaries, and the extraction of "topology" from scanned maps. One interesting application is polygon generalisation: a polygon may be generalised by first constructing its skeleton, then simplifying this skeleton, and finally reconstructing the polygon from the simplified skeleton. This idea has been extended to three dimensions for the reconstruction and simplification of surfaces and features from laser scans (Tam & Heidrich, 2003). The method is similar to the two-dimensional one and involves the extraction of the DT of the set of points, and the 3D skeleton of the reconstructed objects (with the VD). The surface is simplified by modifying the skeleton, which as a consequence involves modifying the crust.

5.2 Modelling 3D Datasets in the Geosciences

In the same way that triangulated irregular networks are used in two dimensions to represent the spatial variation of an attribute (usually the elevation of a terrain), tetrahedra can be used to model geoscientific datasets, as an alternative to raster structures (Lattuada, 1998). Examples are datasets in geology or oceanography where continuous phenomena (e.g. the percentage of gold in the rock or the salinity of the water) are modelled. However, as Ledoux & Gold (2006) explain, modelling 3D continuous phenomena can also benefit greatly from both the DT and the VD. As discussed in Section 2, most of the algorithms and data structures available for constructing and storing the VD actually use only tetrahedra and extract the VD when needed. Although this results in a faster implementation, it has one major drawback if one wants to work with the VD: the computation of the Voronoi elements (vertices, edges, faces and polyhedra) is a computationally expensive operation. Although a single Voronoi vertex is easily extracted (it is located at the centre of the tetrahedron to which it is dual), the extraction of the other elements is more complex. The next examples all refer to Figure 4. A Voronoi edge, which is dual to a triangular face κ, is formed by the two Voronoi vertices dual to the two tetrahedra sharing κ. A Voronoi face, which is dual to a Delaunay edge α, is formed by all the vertices that are dual to the Delaunay tetrahedra incident to α. The idea is simply to "turn" around a Delaunay edge and extract all the Voronoi vertices to extract a face. The construction of one Voronoi cell $\mathcal{V}_p$ dual to a vertex p is similar: it is formed by all the Voronoi vertices dual to the tetrahedra incident to p. Since a Voronoi cell is convex by definition, it is possible to collect all the Voronoi vertices and then compute the convex hull (see Barber et al. (1996) for more details). A simpler method consists of first identifying all the edges incident to p, and then extracting the dual face of each.

In short, in order to model a geoscientific dataset, the DT is an important concept because it helps in the construction and the manipulation of the spatial relationships between the elements, but also because many 3D visualization algorithms are optimised when tetrahedra are involved (Cignoni et al., 1998). On the other hand, the VD can be of great help with many spatial analysis operations, and storing it explicitly considerably optimises the operations. Examples are interpolation methods (e.g. the natural neighbour interpolation method of Sibson (1981)), spatial analysis functions (Okabe et al., 1994), or traditional GIS operations (Gold et al., 1997).

5.3 Modelling Apartment Buildings

Perhaps the main advantage of using the AQE is that attributes can be stored for any elements of both subdivisions (vertices, edges, faces and volumes). Indeed, the vertex attributes may be stored along with the $x - y - z$ coordinates in the vertex; edge attributes in a quad pointing to a vertex; face attributes in a quad pointing to a face; and volume attributes in the vertex that is dual to a volume.

One potential three-dimensional application exploiting duality to store attributes is the modelling of an apartment building. Each room is represented by one 3-cell (see Figure 8 for a simple example), and in a similar way to the example in Figure 2, the dual edge to each face (e.g. the four walls, the floor or the ceiling) is used to store the relationships between two adjacent rooms, such as the visibility (instead of explicitly representing a window for instance), or the noise transmission. Moreover, it is possible to store attributes for the properties of different parts of one room: the finish of a wall or the floor type (attributes attached to 2-cells), the type of moulding (attributes attached to the 1-cells), or the properties for the whole room (e.g. its volume or renting price) can be stored in the dual vertex to the 3-cell. A major advantage of using the AQE for modelling apartment buildings is that a wall (or a floor) is stored twice, once for each of the two rooms sharing the wall. This is consistent with the way one would conceptualise many rooms in a building: the wall separating two rooms will not necessarily have the same properties on each side. For example, the finish on one side of the wall might be different from the one on the other side, or one side might have a gold ceiling moulding and the other no moulding at all.

The AQE can also be used to search efficiently a 3D cell complex. Referring to Figure 8, if q_f (the black quad) is a quad representing the wall between the two rooms, it is possible to retrieve all the elements forming the room on the right by using a simple graph search algorithm such *breadth-first search* (BFS) on the 3-cell representing the room (using quad-edge operators). Each element of the wall that q_f represents can also be retrieved with the quad-edge operators *rot* and *next*; and the same is true for the ceiling or the floor that can be accessed directly from the AQE algebra, without searching. The attributes characterising the relation between the two rooms are obtained with $q_f.through$, and then the attributes of the whole room can be obtained directly because they are stored in the vertex dual to the room. Also, from q_f it is possible to navigate locally and visit the adjacent room with $q_f.adjacent$, and then for instance move to its ceiling by applying *rot* twice. Given a k-cell C, it is actually possible to identify all the l-cells (where $k < l \leq 3$) incident to C using the navigation operators of Section 4.3. Notice that in the case of retrieving all the elements incident to a vertex one can simply move to the dual subdivision and perform a BFS on the 3-cell because the elements retrieved are dual to elements incident in the primal. A further important consideration is that a complete 3D cell complex can be easily and efficiently traversed without storing extra information (de Berg et al., 1997).

When the attributes are included in a search, it is for example possible to look for an edge that represents gold ceiling moulding, incident to a white ceiling and a blue wall, and with a red corner boss; or for all the rooms having more than 100 m^2 but not having a carpet.

6 Discussion

This paper has outlined a new data structure to store simultaneously a 3D cell complex and its dual, and briefly showed its use for different applications. The augmented quad-edge integrates under one structure different types of data structures used in different application domains. While the data structure requires somewhat more storage than some more simple structures, it has the advantage of conceptual simplicity and involves only a simple extension of the 2D topological relationships. The fact that the dual subdivision is also stored permits the storage of attributes for any elements of a 3D subdivision, and makes the structure particularly valuable for a wide range of applications. We believe that the AQE offers a good trade-off between storage costs and the number of topological relationships between the different elements of a subdivision.

We are still exploring if the structure is appropriate for arbitrary subdivisions of 3-manifolds where 'dangling' faces and edges may exist. Future work also includes more formal definition of navigation and construction operations, and possible compression of the structure. Earlier work suggests that quad-edges, being an algebra, may be stored effectively using relational databases, but this is still being examined (Merrett, 2005).

Acknowledgements

Part of this work was done when the authors were at the Hong Kong Polytechnic University, and we thank the support of the Hong Kong's Research Grants

Council (Project PolyU 5068/00E). We would also like to thank Marcin Dzieszko for helping with the programming. Finally, the second author acknowledges the funding of the EU Marie Curie Chair.

References

Amenta, N., Marshall, B., & Eppstein, D. (1998). The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135.

Anselin, L. (1999). Interactive techniques and exploratory spatial data analysis. In P. A. Longley, M. F. Goodchild, D. J. Maguire, & D. W. Rhind, editors, *Geographical Information Systems*, pages 253–266. John Wiley & Sons, second edition.

Aurenhammer, F. (1991). Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405.

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. T. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483.

Baumgart, B. G. (1975). A polyhedron representation for computer vision. In *National Computer Conference*. AFIPS.

Bertrand, Y., Dufourd, J. F., Françon, J., & Lienhardt, P. (1993). Algebraic specification and development in geometric modeling. In *Proceedings TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 75–89. Orsay, France.

Boissonnat, J.-D., Devillers, O., Pion, S., Teillaud, M., & Yvinec, M. (2002). Triangulations in CGAL. *Computational Geometry—Theory and Applications*, 22:5–19.

Boudriault, G. (1987). Topology in the TIGER file. In *Proceedings 8th International Symposium on Computer Assisted Cartography*. Baltimore, USA.

Cignoni, P., Montani, C., & Scopigno, R. (1998). Tetrahedra based volume visualization. In H.-C. Hege & K. Polthier, editors, *Mathematical Visualization—Algorithms, Applications, and Numerics*, pages 3–18. Spinger-Verlag.

de Berg, M., van Kreveld, M., van Oostrum, R., & Overmars, M. (1997). Simple traversal of a subdivision without extra storage. *International Journal of Geographical Information Science*, 11(4):359–374.

De Floriani, L. & Hui, A. (2003). A scalable data structure for three-dimensional non-manifold objects. In *Proceedings 1st Eurographics Symposium on Geometry Processing*, pages 72–82. Aachen, Germany.

Dobkin, D. P. & Laszlo, M. J. (1989). Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32.

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

Gold, C. M. (1993). Forestry spatial decision support system classification, and the 'flight simulator' approach. In *Proceedings GIS'93: Eyes on the Future*, pages 797–802. Vancouver, Canada.

Gold, C. M., Remmele, P. R., & Roos, T. (1997). Voronoi methods in GIS. In M. van Kreveld, J. Nievergelt, T. Roos, & P. Widmayer, editors, *Algorithmic Foundations of Geographic Information Systems*, volume 1340 of *Lecture Notes in Computer Science*, pages 21–35. Springer-Verlag.

Gold, C. M. & Snoeyink, J. (2001). A one-step crust and skeleton extraction algorithm. *Algorithmica*, 30:144–163.

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

Joe, B. (1991). Construction of three-dimensional Delaunay triangulations using local transformations. *Computer Aided Geometric Design*, 8:123–142.

Lattuada, R. (1998). *A triangulation based approach to three dimensional geoscientific modelling*. Ph.D. thesis, Department of Geography, Birkbeck College, University of London, UK.

Ledoux, H. & Gold, C. M. (2006). La modÃ©lisation de donnÃ©es ocÃ©anographiques Ã l'aide du diagramme de VoronoÃ¯ tridimensionnel *(in French). Revue internationale de gÃ©omatique*, 16(1):51–70.

Ledoux, H., Gold, C. M., & Baciu, G. (2005). Flipping to robustly delete a vertex in a Delaunay tetrahedralization. In *Proceedings International Conference on Computational Science and its Applications—ICCSA 2005*, volume 3480 of *Lecture Notes in Computer Science*, pages 737–747. Springer-Verlag, Singapore.

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

Lopes, H. & Tavares, G. (1997). Structural operators for modeling 3-manifolds. In *Proceedings 4th ACM Symposium on Solid Modeling and Applications*, pages 10–18. Atlanta, Georgia, USA.

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

Merrett, T. H. (2005). Topological structures for spatial databases in two and three dimensions. In *Proceedings 17th Australasian Database Conference (ADC2005)*. Newcastle, Australia.

Morehouse, S. (1985). ARC/INFO: A geo-relational model for spatial information. In *Proceedings 7th International Symposium on Computer Assisted Cartography*. Washington DC, USA.

Muller, D. E. & Preparata, F. P. (1978). Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7:217–236.

Okabe, A., Boots, B., & Sugihara, K. (1994). Nearest neighbourhood operations with generalized Voronoi diagrams: A review. *International Journal of Geographical Information Systems*, 8(1):43–71.

Requicha, A. A. G. (1982). Representation of rigid solids—theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260.

Shewchuk, J. R. (1997). *Delaunay Refinement Mesh Generation*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburg, USA.

Sibson, R. (1981). A brief description of natural neighbour interpolation. In V. Barnett, editor, *Interpreting Multivariate Data*, pages 21–36. Wiley, New York, USA.

Tam, R. & Heidrich, W. (2003). Shape simplification based on the medial axis transform. In *Proceedings IEEE Visualization 2003*, pages 481–488. IEEE Computer Society, Seattle, Washington, USA.

Zlatanova, S., Abdul Rahman, A., & Shi, W. (2004). Topological models and frameworks for 3D spatial objects. *Computers & Geosciences*, 30:419–428.

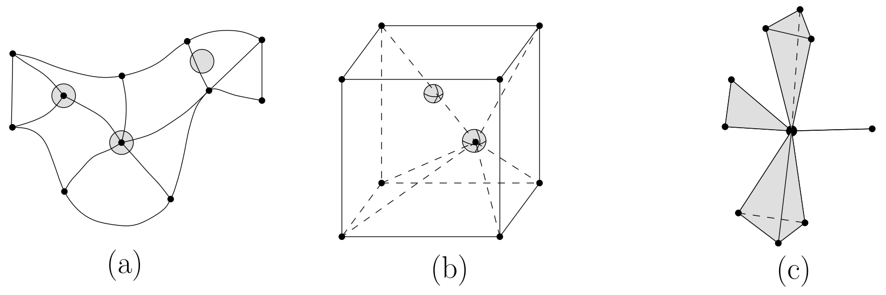

Figure 1: **(a)** The neighbourhood of every point in a 2-manifold is equivalent to a plane. **(b)** The neighbourhood of every point in a 3-manifold is equivalent to a 3D space. **(c)** A non-manifold object. The neighbourhood of the vertex where the two tetrahedra, the triangular face and the edge meet does not resemble a 3D space, i.e. a coordinate system can not be defined locally.

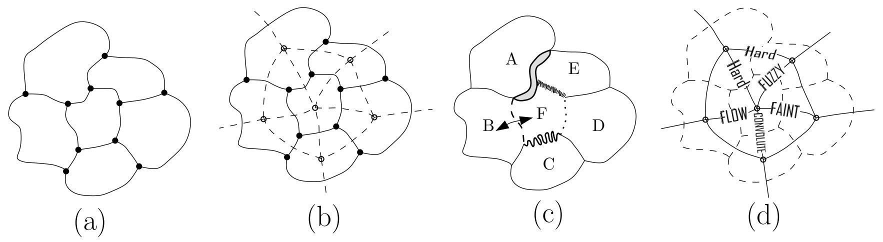

Figure 2: The dual graph is used to describe the relationships between adjacent polygons.

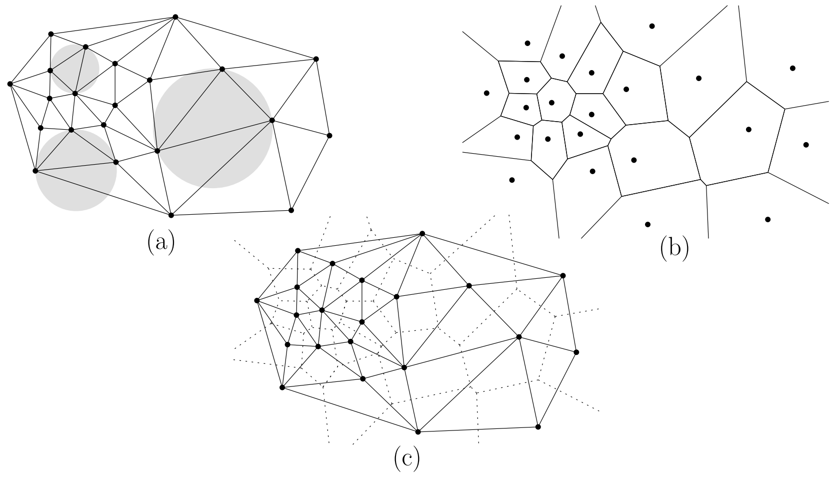

Figure 3: **(a)** The Delaunay triangulation for a set of points in the plane. **(b)** The Voronoi diagram for the same set of points. **(c)** Both structures.

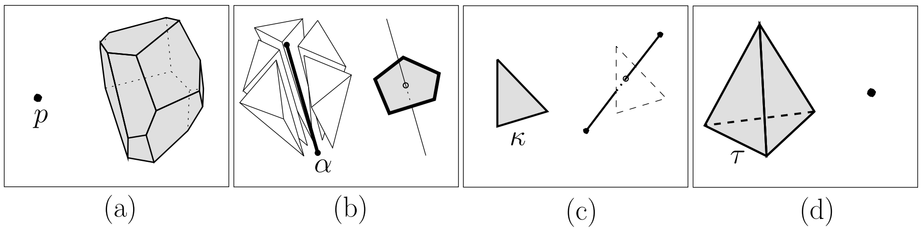

Figure 4: Duality in 3D between the elements of the Voronoi diagram and the Delaunay tetrahedralization.

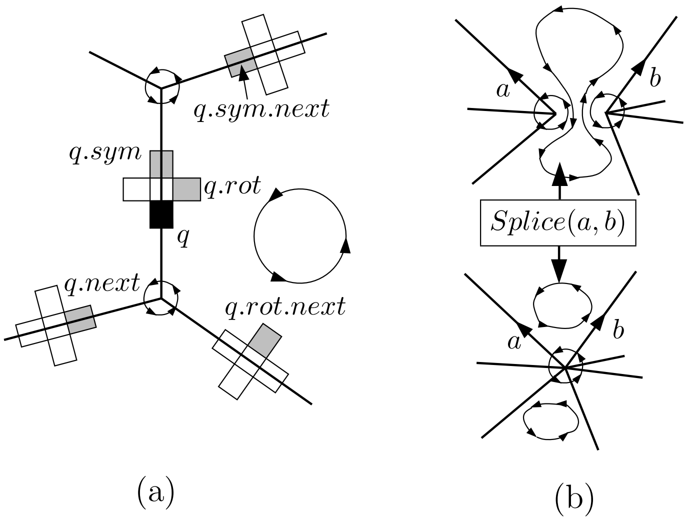

Figure 5: **(a)** The quad-edge data structure and some basic operators (only one subdivision is represented). The starting quad q is the black quad, and the resulting quads are grey. **(b)** The *Splice* operator.

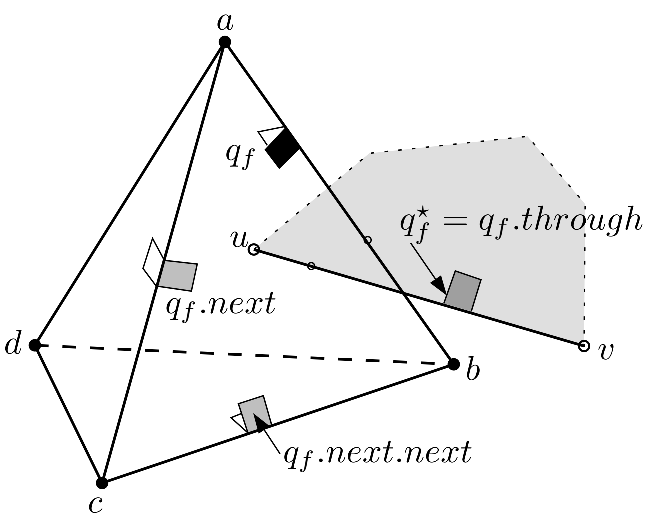

Figure 6: The *through* pointer. The quad q_f is the starting quad, and the grey ones are resulting quads after an operation.

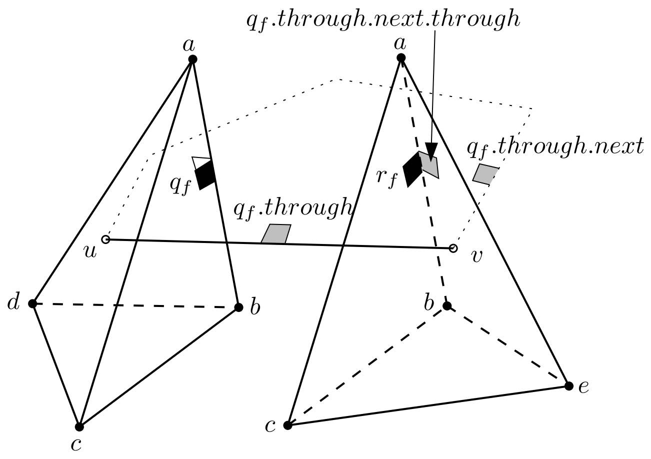

Figure 7: The *adjacent* operator. The quads q_f and r_f represent the same pair (face, edge), but belong to different 3-cells.

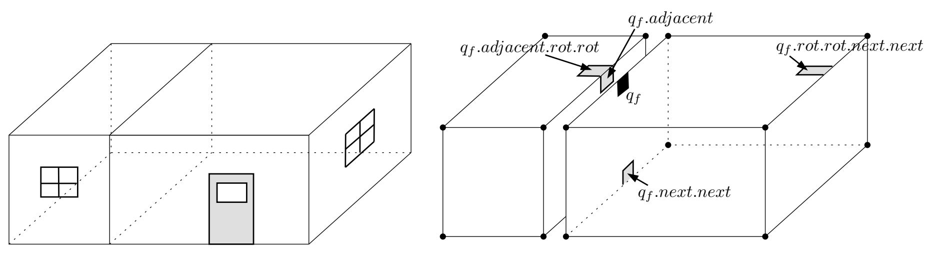

Figure 8: An apartment building can be modelled with the AQE.