# AUGMENTED QUAD-EDGE – 3D DATA STRUCTURE FOR MODELLING OF BUILDING INTERIORS

P. Boguslawski [a], C. Gold [b]

[a] Faculty of Advanced Technology, University of Glamorgan, Pontypridd CF37 1DL UK - pbogusla@glam.ac.uk
[b] Faculty of Advanced Technology, University of Glamorgan, Pontypridd CF37 1DL UK - cmgold@glam.ac.uk

**KEY WORDS:** data structure, three-dimensional modelling, duality, Voronoi diagram, Delaunay tetrahedralization

**ABSTRACT:**

This work presents a new approach towards the construction and manipulation of 3D cells complexes, stored in the *Augmented Quad-Edge* (AQE) data structure. Each cell of a complex is constructed using the usual *Quad-Edge* structure, and the cells are then linked together by the dual edge that penetrates the face shared by two cells.
We developed a new set of atomic operators that allow for a significant improvement of the related construction and navigation algorithms in terms of computational complexity. The idea is based on simultaneous construction of both the 3D Voronoi Diagram and its dual the Delaunay Triangulation.
We expect that the increase of the efficiency related to the simultaneous manipulation of the both duals will allow for many new applications, like real-time analysis and simulation of modelled structures.

## 1. INTRODUCTION

The Voronoi diagram (VD) and the Delaunay triangulation/tetrahedralization (DT) can be used for modelling different kinds of data for different purposes. They can be used to represent the boundaries of real-world features, for example geological modelling of strata or models of apartment buildings. The VD and the DT are dual – they represent the same thing from a different point of view – and both structures have interesting properties (Aurenhammer, 1991).

The Delaunay triangulation of the set of points in two-dimensional Euclidean space is the triangulation of the point set with the property that no point falls in the interior of the circumcircle of any triangle in the triangulation. If we connect the centres of these circles between pairs of adjacent triangles we get the Voronoi diagram, the dual of the Delaunay triangulation, with one Voronoi edge associated with each Delaunay edge. The Voronoi diagram consists of cells around the data points such that any location in a particular cell is closer to its cell generating point than to any other (Mostafavi, et al., 2003).

Most of the algorithms and implementations available to construct the 3D VD/DT store only the DT, and if needed the VD is extracted afterwards. This has major drawbacks if one wants to work with the VD. It is for example impossible to assign attributes to Voronoi vertices or faces. In many applications, the major constraint is not the speed of construction of the topological models of large number of number of points, 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.

The 2D case has already been solved with the *Quad-Edge* data structures of Guibas and Stolfi (1985). The structure permits the storage of any primal and dual subdivisions of a two-dimensional manifold. Dobkin and Laszlo (1989) have generalized the ideas behind the *Quad-Edge* structure to preserve the primal and dual subdivisions of a three-dimensional manifold. Their structure, the *Facet-Edge*, comes with an algebra to navigate through a subdivision and with primitives construction operators. 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. Other data structures (see (Lienhardt, 1994), (Lopes and Tavares, 1997)) can usually store only one subdivision.

## 2. THE QUAD-EDGE DATA STRUCTURE

The *Quad-Edge* as a representation of one geometrical edge consists of four *quads* which point to two vertices of an edge and two neighbouring faces. It allows navigation from edge to edge of a connected graph embedded in a 2-manifold. Its advantages are firstly that there is no distinction between the primal and the dual representations, and secondly that all operations are performed as pointer operations only, thus giving an algebraic representation to its operations. Figure 1 shows the basic structure and navigation operators (*next*, *rot* and *sym*).

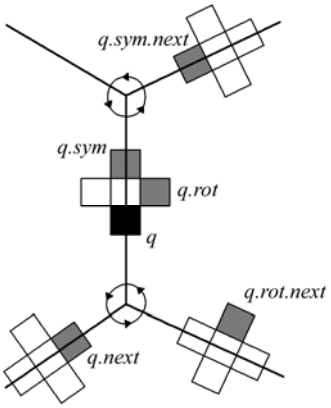


Figure 1. The *Quad-Edge* structure and basic operators: *rot*, *sym*, *next* (Ledoux, 2006)

## 3. AUGMENTED QUAD-EDGE (AQE)

The AQE (Ledoux and Gold, in press), (Gold, et al., 2005) uses the *Quad-Edge* to represent each 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 a boundary representation. 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 complex two things are missing: a ways to link adjacent cells in a given space, and also a mechanism to navigate to the dual space. In this case two of the four *org* pointers are not used in 3D. The idea is to use the free face pointers in the *Quad-Edge* to link two cells sharing a face. 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* to a *Quad-Edge* in the dual cell complex, and from there we may return to a different cell in the original cell complex.

The AQE is high in storage but it is computationally efficient (Ledoux, 2006). Each tetrahedron contains 6 edges – each one is represented by four quads containing 3 pointers. This makes a total of 72 pointers. The total number of pointers for the dual is also 72. It makes a total of 144 pointers for each tetrahedron. However we preserve valuable properties which are crucial in real-time computations.

**Construction and navigation** In previous work the theoretical basis of the storage and manipulation of 3D subdivisions with use of the AQE were described (Ledoux and Gold, in press) and it was shown that this model worked.

The main construction operator is *MakeEdge*. It creates a single edge, that at the moment of creation it is not linked to any other edge. The *Splice* operator is used to link edges in the same subdivision. Edges in the dual subdivisions are linked one-by-one later using the *through* pointer.

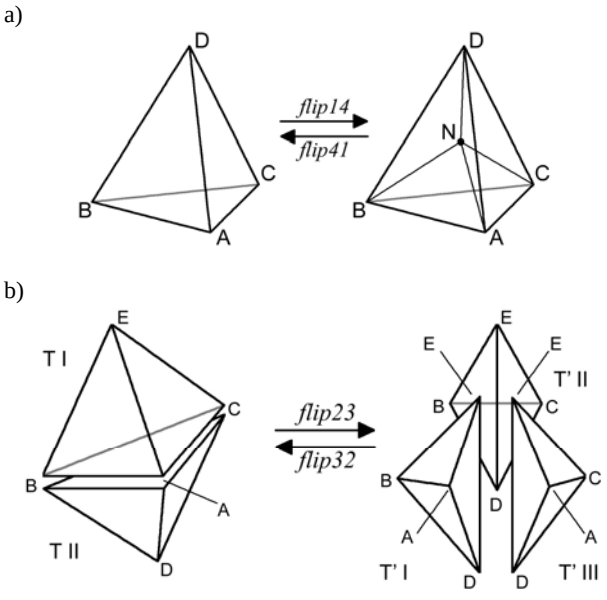


Figure 2. *Flip* operators (Ledoux, 2006): a) *flip14* is used when a new point is inserted. Its reverse is *flip41*; b) *flip23* is used when the structure has to be modified in order to preserve the correct DT. Its reverse is *flip32*.

When a new point is inserted in the structure of the DT, four new tetrahedra are created inside the already existing one that contains the newly inserted point. Then the enclosing tetrahedron is removed. The new corresponding Voronoi points are calculated and another tetrahedron is created separately in the dual subdivision. Then all edges are linked together and, to maintain a properly built DT structure, subdivisions are modified by *flip* operators. Two basic flip operators are shown in Figure 2.

Another requirement for the navigation is the *through* pointer that links together both dual subdivisions (Ledoux and Gold, in press), (Ledoux, 2006). The *org* pointers that are not used in 3D allow for making a connection to the dual edge. With this operator it is possible to go to the dual subdivision and back to the origin. It is the only way to connect two different cells in the same subdivision.

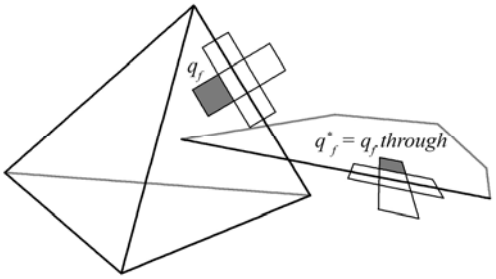


Figure 3. The *through* pointer is used to connect both dual subdivisions (Ledoux, 2006)

To get the shared face of two cells, the adjacent operator is used. It is a complex operator that consists of a sequence of *through* and other basic operators. (Ledoux, 2006)

## 4. ATOMIC OPERATORS

The general algorithm of the point insertion to the DT/VD structure was described by Ledoux (2006). In our current work we have implemented and improved the way of building the whole structure.

Algorithm 1: *ComplexMakeEdge*(DOrg, DDest, VOrg, VDest)
*// DOrg, DDest – points defined edge in DT*
*// VOrg, VDest – points defined edge in VD*

e1:=MakeEdge(DOrg, DDest);
e2:=MakeEdge(VOrg, VDest);
e1.Rot.V:=e2;
e2.InvRot.V:=e1.Sym;

The most fundamental operator is *ComplexMakeEdge* which creates two edges using *MakeEdge* (Ledoux, 2006). They are dual and the one belongs to the DT and the second to the VD. The V pointer from the *Quad-Edge* structure is used to link them as shown in Algorithm 1. We claim that the connection between the newly created edges in both dual subdivisions has a very important property – it is permanent and not changed by any other operator.

Algorithm 2: *InsertNewPoint*(N) – *ComplexFflip14*
// N – new point inserted to the DT

1. Find tetrahedron which contain point N
2. Calculate 4 new Voronoi points
3. Create new edges with using *ComplexMakeEdge* with point N and new Voronoi points
4. Assign *through* pointers
5. Disconnect origin edges of tetrahedron using *Splice*
6. Connect edges of 4 new tetrahedra using *Splice*
7. Add 4 new tetrahedra to a stack
8. while necessary do *flip23* or *flip32* for tetrahedra from the stack

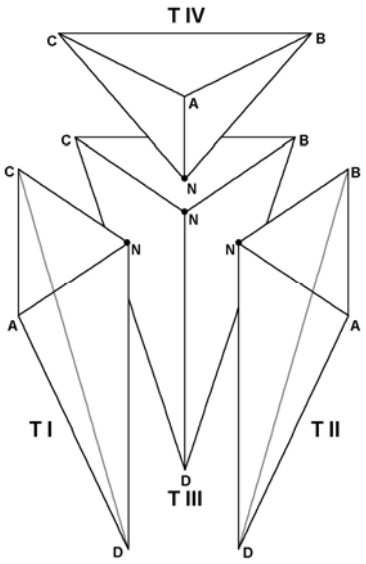


Figure 4. *fli*p14 divides origin tetrahedron ABCD into 4 new

The first operation in the point insertion to the structure is *flip14* (Fig. 2a). It divides space occupied by tetrahedron ABCD into four smaller ones (Fig. 4). The inserted point N is a vertex shared by new tetrahedra. As mentioned above, this version of the algorithm is an improvement over Ledoux (2006). The significant aspect is that we don't remove the origin tetrahedra and create 4 new. Edges from the origin tetrahedron are disconnected and used to create 4 new. Thus no edges are deleted from the DT structure. What is more, the same applies to the VD because dual edges are linked together permanently. Only new edges are added to the structure.

| Tetra-hedron | Edges from origin ABCD tetrahedron used to create 4 new | Newly created edges |
|---|---|---|
| T I | CA, AD | DC, CN, AN, DN |
| T II | AB, BD | DA, AN, BN, DN |
| T III | BC, CD | DB, BN, CN, DN |
| T IV | - | BA, AC, CB, BN, AN, CN |

Table 5. Edges used in *flip14*

Table 5 in conjunction with Fig. 2a) shows which edges are created and which ones are taken from the origin tetrahedron.

The operation of point insertion does not demand any modification to the whole structure except for local changes of a single cell. This case is implemented in the *ComplexFlip14* operator (Algorithm 2). The structure created this way keeps all new cells connected, and navigation between them, and within the whole structure, remains possible. The new complex operator is more efficient because it requires fewer operations to insert a point and modify the structure.

| Tetra-hedron | Edges from origin tetrahedra used to create new ones | Newly created edges | Deleted edges |
|---|---|---|---|
| T' I | from TI: BE<br>from TII: BD, AB | AE, AD, DE | AB (from TI) |
| T' II | from TI: AE<br>from TII: AD, CA | CE, CD, DE | CA (from TI) |
| T' III | from TI: CE<br>from TII: CD, BC | BE, BD, DE | BC (from TI) |

Table 6. Edges used in *flip23*

Algorithm 3: *flip23(TI, TII)*:
*// TI, TII – two adjacent tetrahedra*

1. Calculate 3 new Voronoi points
2. Copy edges and create new ones as shown in Table 6
3. Assign *through* pointers
4. Disconnect edges of 2 original tetrahedra using *Splice*
5. Connect edges of 3 new tetrahedra using *Splice*
6. Remove spare edges (see Table 6)
7. Remove 2 old Voronoi points
8. Add 6 new tetrahedra to the stack

| Tetra-hedron | Edges from origin tetrahedra used to create new ones | Newly created edges | Deleted edges |
|---|---|---|---|
| T I | from T'I: BE<br>from T'II: CE<br>from T'III:AE | AB, BC, CA | from T'I: AE, AD, DE<br>from T'II: BE, BD, DE<br>from T'III: CE, CD, DE |
| T II | from T'I: AB, BD<br>from T'II: BC, CD<br>from T'III: CA, AD | - | |

Table 7. Edges used in *flip32*

Algorithm 4: *flip32*(TI, TII, TIII):
// TI, TII, TIII – three tetrahedra adjacent in pairs

1. Calculate 2 new Voronoi points
2. Copy some edges and create new ones as shown in Table 7
3. Assign t*hrough* pointers
4. Disconnect edges of 3 origin tetrahedra using *Splice*
5. Connect edges of 2 new tetrahedra using *Splice*
6. Remove spare edges (see Table 7)
7. Remove 3 old Voronoi points
8. Add 6 new tetrahedra to the stack

Finally all edges are linked together to give a correctly built structure. Then correctness tests are performed. They check if the new tetrahedra have built the correct DT structure. If not, *flip23* (Algorithm 3) or *flip32* (Algorithm 4) are executed

(Ledoux, 2006). Edges taking part in these operators are listed in Tables 6 and 7 and showed in Fig. 2b).
To check the validity of our assumptions a special computer application was created. The implementation showed that our new complex operators work. The number of required operations for creation and deletion of edges and assignment of pointers has significantly decreased from the previous work of (Gold, et al., 2005).

## 5. COMPUTER AIDED MODELLING

Emergency planning and design of buildings are major issues for many people especially after 11th September 2001. To manage disasters effectively they need tools for rapid building plan compilation, editing and analysis.

In many cases 2D analysis is inadequate for modelling building interiors and escape routes. 3D methods are needed. This is more obvious in disciplines such as geology (with complex adjacencies between rock types) and building construction (with security aspects). There is no appropriate data structure to describe those issues in a "3D GIS" context.

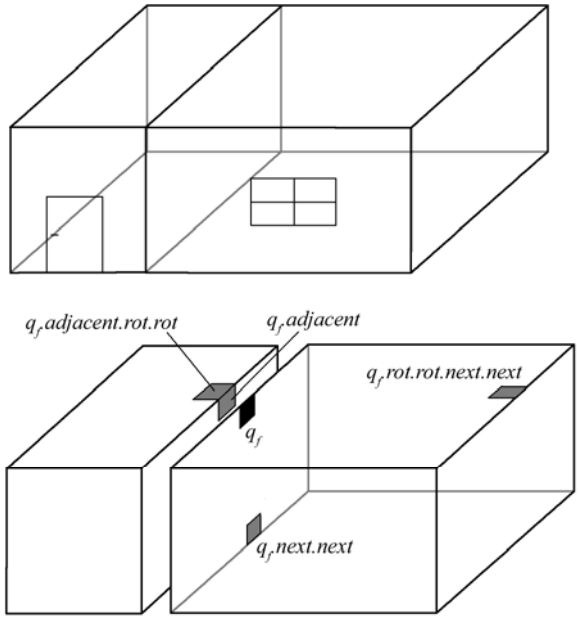


Figure 8. The AQE is an appropriate structure for the modelling of building interiors. (Ledoux, 2006)

The new operators can be used for advanced 3D modelling. In our opinion the AQE is a good structure for the modelling of building interiors (Fig. 8). Faces in the structure are stored twice, so every wall separating two rooms can have different properties on each side. It can help to make models not only of simple buildings but also of overpasses, tunnels and other awkward objects. It will be possible to create systems for disaster management, for example to simulate such phenomena as spreading fire inside buildings, flooding, falling walls, terrorist activity, etc.

Another example is navigation in buildings, which requires the primal graph for forming rooms and the dual graph for making connections between rooms. Even though one can be reconstructed from the other, they both are needed for full real-time query and editing. These graphs need to be modifiable in real-time to take account of changing scenarios. This 3D Data Structure will assist applications in looking for escape routes from buildings.

## 6. CONCLUSIONS

Our current work involved the development and improvement of the atomic construction operations similar to the *Quad-Edge*. When we complete all atomic operators and prove their correctness, we will be able to use binary operations for location of quads in the stored structures. That will improve the efficiency of algorithms and allow for their use in real-time applications.

In future work we will try to create a basic program for the modelling of building interiors and implement new functions such as the evaluation of optimal escape routes. We believe that such basis "edge algebra" has many practical advantages, and that it will be a base for many future applications.

## REFERENCES

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

Dobkin, D. P. and Laszlo, M. J., 1989. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4, pp. 3-32.

Gold, C. M., Ledoux, H. and Dzieszko, M., 2005. A Data Structure for the Construction and Navigation of 3D Voronoi and Delaunay Cell Complexes. *WSCG'2005 Conference*, Plzen, Czech Republic.

Guibas, L. J. and Stolfi, J., 1985. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4, pp. 74-123.

Ledoux, H. and Gold, C. M., in press. Simultaneous storage of primal and dual three-dimensional subdivisions. *Computers, Environment and Urban Systems.*

Ledoux, H., 2006. Modelling three-dimensional fields in geoscience with the Voronoi diagram and its dual. Ph.D. dissertation, School of Computing, University of Glamorgan, Pontypridd, Wales, UK.

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

Lopes, H. and Tavares, G., 1997. Structural operators for modelling 3-manifolds. *Proceedings 4th ACM Symposium on Solid Modeling and Applications*, Atlanta, Georgia, USA, pp. 10-18.

Mostafavi, M. A., Gold, C. M. and Dakowicz, M., 2003. A Delete and insert operations in Voronoi/Delaunay methods and applications. *Computers & Geosciences*, 29 (4), pp. 523-530.