A Data Structure for the Construction and Navigation of 3D Voronoi and Delaunay Cell Complexes

Christopher Gold[1] christophergold@voronoi.com
Hugo Ledoux[2] hugo.ledoux@gmail.com
Marcin Dzieszko[2] marcindzieszko@wp.pl

[1] School of Computing, University of Glamorgan, Pontypridd CF37 1DL, Wales, UK
[2] Department of Land Surveying & Geo-Informatics, Hong Kong Polytechnic University, Hong Kong.

1 INTRODUCTION

The Voronoi diagram (VD) and the Delaunay triangulation (DT) can be used for modelling different kinds of data for different purposes. These two structures are attractive alternatives to rasters to discretise a continuous phenomenon such as the percentage of gold in the soil, the temperature of the water, or the elevation of a terrain. They can also be used to represent the boundaries of real-world features, for example geological modelling of strata or cadastral models of apartment buildings. The VD and the DT are an appealing solution because of their duality (they represent the same thing, just from a different point of view) and because both structures have interesting properties (see Aurenhammer [Aur91] for a review of the properties and potential applications).

Most of the algorithms and implementations available to construct the three-dimensional VD/DT store only the DT and perform their topological operations on tetrahedra, and if needed the VD is extracted afterwards. Although this results in a faster implementation, it has major drawbacks if one wants to work with the VD. It is for example impossible to assign attributes to Voronoi vertices or faces, and moreover the computation (extraction) of the VD is an expensive operation. We believe that in many real-world applications, the major constraint is not the speed of construction of the topological models of large 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. We also think that there are many reasons that justify preserving simultaneously both the VD and the DT. The two-dimensional case has already been elegantly solved with the *quad-edge* data structures of Guibas and Stolfi [GS85]. The structure permits the storage of any primal and dual subdivisions of a two-dimensional manifold. Dobkin and Laszlo [DL89] 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*, as is the case for the quad-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 and, to our knowledge, has not been used in 'real projects'. Other data structures, e.g. see [Lie94, LT97], can usually store only one subdivision.

To store and manipulate three-dimensional cells complexes, we propose a simpler structure, based on the quad-edge, that we name *augmented quad-edge* (AQE). Each cell of a complex is constructed using the usual quad-edge structure, and the cells are linked together by the dual edge that penetrates the face shared by two cells. When some basic navigation and construction operators are added to the structure, it is possible to construct and store simultaneously the 3D VD and its dual the DT. While using somewhat more storage, the approach has the advantage of conceptual simplicity and involves only a simple extension of the 2D topological relationships.

POSTERS proceedings ISBN80-903100-8-7
WSCG'2005, January 31–February 4, 2005
Plzen, Czech Republic.

2 AUGMENTED QUAD-EDGE

The quad-edge data structure [GS85] 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 be-

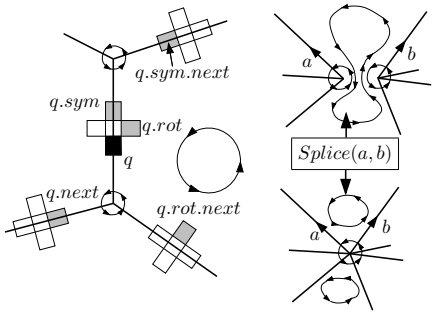

Figure 1: Left: The quad-edge structure and some basic operators. The starting quad q is the black quad, and the resulting quads are grey. Right: The *Splice* operator.

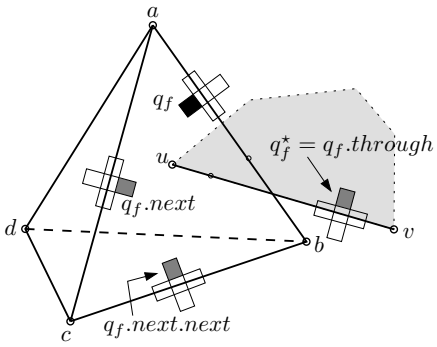

Figure 2: The *through* pointer.

tween the primal and the dual representation, 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, the different navigation operators (*next*,*rot* and *org*), and the construction operators *Splice* and *MakeEdge*.

The augmented quad-edge (AQE) uses the 'normal' quad-edge, which is valid for any 2-manifolds, 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 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 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. 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. Notice 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 Dobkin and Laszlo's structure [DL89], a pair (face, edge). Therefore q_f has also a linked quad $q_f^\star$ in the dual that is defined by (edge*, face*).

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* (Figure 2). 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.

Acknowledgements. We would like to thank the support from Hong Kong's Research Grants Council, Hong Kong (Project PolyU 5068/00E and through a research studentship to the second author).

References

[Aur91] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[DL89] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.

[GS85] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4:74–123, 1985.

[Lie94] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324, 1994.

[LT97] H. Lopes and G. Tavares. Structural operators for modeling 3-manifolds. In *Proceedings 4th ACM Symposium on Solid Modeling and Applications*, pages 10–18, Atlanta, Georgia, USA, 1997.