# MODELLING OCEANOGRAPHIC DATA WITH THE THREE-DIMENSIONAL VORONOI DIAGRAM

Hugo Ledoux and Christopher Gold

Dept. Land Surveying & Geo-Informatics, Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
hugo.ledoux@polyu.edu.hk, christophergold@voronoi.com

**KEY WORDS:** GIS, Three-dimensional, Modeling, Oceanography, Triangulation, Algorithms, Data Structures, Dynamic

**ABSTRACT:**

Managing oceanographic data with traditional geographical information systems (GIS) is a difficult task because these systems have been primarily designed for land-based applications. The main problem is that the nature of objects at sea is completely different from the nature of objects found on the land: at sea most objects are represented by unconnected points that can have three-dimensional coordinates, the datasets have 'abnormal' distribution and the objects tend to change position over time. We propose in this paper using a spatial model based on the three-dimensional Voronoi diagram (VD) to handle topological relationships between objects. We present the main properties of the 3D VD, algorithms to construct and modify it, and show how some 3D GIS operations are greatly simplified when a spatial model is built upon it

## 1. INTRODUCTION

Data collected for marine applications have particular properties that are usually not present in data collected on the land. First, because almost no man-made objects are found at sea, the objects (samples) are mostly represented by unconnected points, to which some attributes are attached. Second, the samples are usually collected from a boat, which results in datasets having highly irregular distribution (samples are distributed according to each ship's track). Two-dimensional datasets (e.g. bathymetric samples having *x-y* coordinates and depth of water) are very difficult to manage with traditional geographical information systems (GIS) because their spatial model is built for two-dimensional land applications and their data structure is based on the 'overlays' as a definition of adjacencies between objects (Gold and Condal, 1995). Three-dimensional oceanographic datasets are usually composed of CTD data: attributes (Conductivity-Temperature-Depth) of the water are measured with a sensor that is moved through the water column. A three-dimensional (volumetric) representation of the water is built with many water columns collected along different ship tracks. Samplings obtained in such a way are sparse in the horizontal direction but abundant in the vertical direction. The integration of such datasets into traditional GIS is problematic because these systems usually deal only with surfaces and two-dimensional objects, and, as a result, datasets must often be 'reduced' by one dimension (for example by 'slicing' it) to be integrated and analysed. Some solutions exist – using 3D raster data structures as in the work of Jones (1989), Raper (1989) and O'Conaill et al. (1992) – but, as shown in Section 4 , they have shortcomings for oceanographic data. A further important consideration is that the marine environment is dynamic, which means that objects are likely to move over time.

The many problems arising when using a traditional GIS for handling marine data have been described by many researchers (Davis, 1988; Li, 1993; Lockwood, 1995). Using a spatial model based on the two-dimensional Voronoi diagram (VD), as Gold and Condal (1995) propose, solves most of the problems mentioned earlier. As explained in Section 2, the VD will adapt naturally to the distribution of the data and its 'tiling' properties can be used to manage the topological relationships between unconnected objects. Moreover, unlike the structure of traditional GIS, the topology can be updated locally. Wright and Goodchild (1997), in a review, affirmed that this method was the only published attempt at that time to solve many important problems related to the nature of marine data. The only problem not tackled by Gold and Condal is 3D volume-based representations.

In this paper, we extend the work of Gold and Condal (1995) and propose using the Voronoi diagram in three dimensions to handle the topological relationships in oceanographic datasets. As shown in Section 2, the concepts and properties of the VD can all be generalized to three dimensions, and, as a result, we have a spatial model capable of solving most of the problems we have when dealing with oceanographic data. Although the concepts easily generalize, their implementations are not straightforward. For this reason, we discuss in Section 3 the main construction and modification algorithms, and also different data structures for storing the VD and its geometric dual, the Delaunay tetrahedralization (DT). As described in Section 4, such a spatial model has numerous advantages over other knows methods. One of them is that many three-dimensional spatial analysis operations are greatly simplified and optimised, and we show in Section 5 how some of these operations, when applied to an oceanographic dataset, can help us to have a better understanding of it.

## 2. PROPERTIES OF THE 3D VORONOI DIAGRAM

The Voronoi diagram for a set of points in a given space $R^d$ is the partitioning of that space into regions such that all locations within any one region are closer to the generating point than to any other. In two dimensions, each cell around a data point is a convex polygon, having a defined number of neighbours; for example in Figure 1 the point *p* has 7 neighbouring Voronoi cells. In three dimensions, a Voronoi cell generalizes to a convex polyhedron formed by convex faces, as shown in Figure 2. In any dimensions, the VD has a geometric dual structure called the Delaunay triangulation. In 2D, this structure is defined by the partitioning of the plane into triangles – where the vertices of the triangles are the points generating each Voronoi cell – that satisfy the *empty circumcircle* test (a circle is *empty* when no points is in its interior, but more than three points can be directly on the circle). The two-dimensional DT is illustrated in Figure 1 by the dashed lines. The Delaunay triangulation is popular for modelling surfaces because among all the possible triangulations of a set of points, it creates one where the minimum angle in each triangle is maximized (triangles are as equilateral as possible), thus being useful for interpolation. The generalization to three dimensions of the Delaunay triangulation is the Delaunay tetrahedralization: each triangle becomes a tetrahedron that satisfies the *empty circumsphere* rule. The DT is unique for a set of points, except when there are degenerate cases in the set (if five or more points are *cospherical* in 3D). In these cases, an arbitrary choice must be made among all the possible solutions. The number of tetrahedra in a DT constructed with $n$ points depends on the configuration of these points, and can be up to $O(n^2)$.

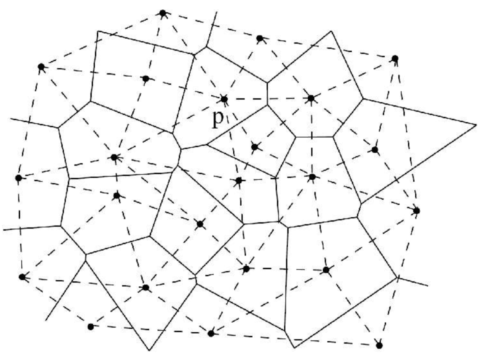


Figure 1. Two-dimensional VD (bold lines) and DT (dashed lines).

Most of the properties of the 2D VD/DT generalize to 3D, except that the minimum angle in each Delaunay tetrahedra is not maximized. There can indeed be almost 'flat' Delaunay tetrahedra. These tetrahedra, called *slivers*, have their four vertices almost lying on a plane and thus have a volume of nearly zero. For many applications where the Delaunay tetrahedralization is used, e.g. to perform simulation in engineering or when the tetrahedra are used to perform interpolation directly, these tetrahedra are bad and must be removed. Here, one might wonder why use them if their properties are not good? First, it should be said that in most cases the Delaunay tetrahedralization has a tendency to favour equilateral tetrahedra over slivers. Second, the Voronoi diagram is not affected by them; the Voronoi cells in 3D will still be 'round' (i.e. relatively spherical) even if the DT has many slivers. Third, many GIS operations (e.g. spatial analysis functions) use the properties of the VD, and if only one tetrahedron is not Delaunay, then the VD is corrupted.

Both the VD and the DT represent the same thing, just from a different viewpoint. The duality between the two structures in three dimensions is simple: each polyhedron becomes a point and each line becomes a face, and vice-versa. For example, a

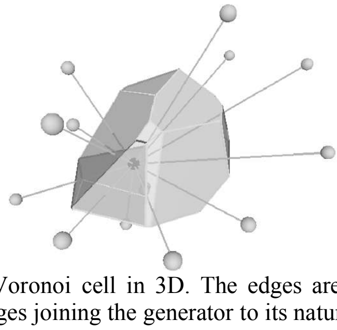

Figure 2. A Voronoi cell in 3D. The edges are the Delaunay edges joining the generator to its natural neighbours.

Delaunay tetrahedron becomes a Voronoi vertex (its position is the centre of the *circumsphere* around the tetrahedron); a Delaunay edge becomes a (convex) Voronoi face; and a Delaunay triangular face becomes an edge spanned by the two Voronoi vertices that are dual to the two tetrahedra sharing the face. For example, in Figure 2, the number of edges joining the generator is equal to the number of faces of the Voronoi cell.

## 3. 3D VD/DT ALGORITHMS AND DATA STRUCTURES

As mentioned in the previous section, both the VD and the DT are geometrically equivalent. By having one structure, its dual can always be constructed. Because it is easier, from an algorithmic and data structure point-of-view, to manage tetrahedra over arbitrary polyhedra (they have a constant number of vertices and neighbours), we construct, store and modify a VD by working only on its dual. The VD is extracted from a DT in $O(n)$ time, $n$ being the number of data points in the set.

We first describe in this section basic operations needed to construct and modify a Delaunay tetrahedralization and then discuss some possible data structures that can be used to efficiently store the DT and/or the VD.

### 3.1 Flipping in 3D

A flip is a local topological operation that modifies the configuration of adjacent tetrahedra in a tetrahedralization. If we consider five points $\{a, b, c, d, e\}$ in $R^3$, there exist three ways to tetrahedralize them: either with two, three or four tetrahedra, depending on their configuration in space. Figure 3 shows one such configuration: the point $e$ is inside a tetrahedron $abcd$. Figure 4 shows the other configuration where the polyhedron $abcde$ is tetrahedralized with either 2 or 3 tetrahedra. Based on this, we can define different kinds of flips. A flip14 is the operation that will insert $e$ inside a tetrahedron $abcd$ (splitting it into 4 tetrahedra), and a flip41 is the inverse operation that will delete $e$ and merge together the 4 tetrahedra. A flip23 transforms a tetrahedralization of 2 tetrahedra by one with 3, and a flip32 is the inverse operation.

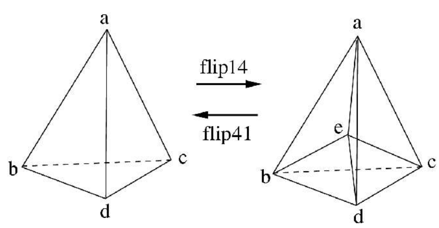


Figure 3. Flips 14 and 41.

### 3.2 Point Location

The point location problem involves finding which tetrahedron in a DT contains a query point $x$. This is needed for different operations, for example to insert a new point in a DT or to interpolate, as it is explained in Section 5. The method we describe here, called 'walking', was discussed in the earliest papers about the construction of triangulations in two dimensions (Gold et al., 1977; Green and Sibson, 1978).

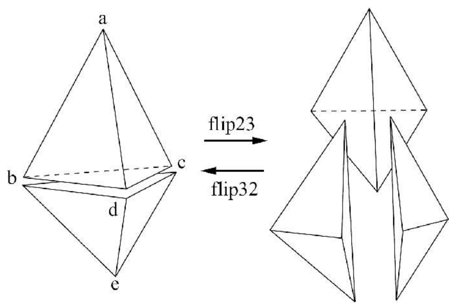

Figure 4. Flips 23 and 32.

Its generalization to three dimensions is straightforward as the method uses only the adjacency relationships between the triangles. The idea is: starting from a given tetrahedron $t$, we move to one of its neighbours $t_l$ if the query point $x$ and $t_l$ are on the same side of the triangular face shared by $t$ and $t_l$. We continue walking from tetrahedron to tetrahedron until $t_l$ has no such neighbour, which means that $t_l$ contains $x$. The method is simple to implement as only one function – one that determines if a point is left or right of a plane in 3D – is needed and no extra storage is required. It is also very efficient in practice, as Mücke et al. (1999) show.

### 3.3 Construction Algorithms

Many different algorithms can be used to construct a 3D VD. One solution, as described in Brown (1979), involves firstly constructing the convex hull of the set of points in ($d$ + 1) dimensions – 4D in our case – and then projecting the result one dimension lower to get the Delaunay tetrahedralization. Implementations of convex hull algorithms in higher dimensions are readily available, e.g. *Qhull* (Barber et al., 1996). Another solution is using the *DeWall* algorithm (Cignoni et al., 1998), which is based on the divide-and-conquer paradigm. These algorithms might be useful for the construction of a DT, but local modifications (insertion of a new point, deletion or movement of one) are either slow and complicated, or simply impossible.

Algorithms that allow local modifications are called 'incremental insertion algorithms' and they proceed as follow to construct a DT. Starting from a valid DT, each point of a set is added one at a time and the tetrahedralization is updated after each insertion. To insert a single point $x$ in a DT, the following steps are required. First, the tetrahedron that contains $x$ must be identified with the point location algorithm described in the previous section. Then, all the tetrahedra whose *circumspheres* contain $x$ must be deleted and replaced by new ones. The first increment insertion algorithm, valid in any dimensions, was developed by Watson (1981). His idea is simple: all the tetrahedra that 'conflict' with $x$ are deleted from the DT and then the hole thus created is filled by creating edges linking $x$ to every vertex of the hole (they prove that the new resulting tetrahedra are guaranteed to be Delaunay). Although the algorithm is simple to implement, the fact that the tetrahedralization is temporarily destroyed can corrupt the algorithm. Field (1986) explains the problems that are encountered when implementing the method.

Another incremental insertion algorithm, due to Joe (1991), is numerically more stable because a complete tetrahedralization is kept during the whole updating process. The conflicting tetrahedra are deleted and replaced by new ones by a sequence of flips. The first step is the insertion of $x$ in the tetrahedron that contains it by using a flip14. The four new tetrahedra are then added to a stack that will control the sequence of flips to perform to restore the 'Delaunayness' in the tetrahedralization. Each tetrahedron on the stack must be tested against its neighbours, if it is not Delaunay then a flip – a flip23 or a flip32, depending on the configuration of adjacent tetrahedra – will destroy some tetrahedra and replace them by other ones (the new ones are then pushed on the stack). The algorithm terminates when the stack is empty. The time complexity of this algorithm, and of Watson's algorithm, is O($n^2$) for a set of $n$ points, not just for the insertion of a single point. This is worst-case optimal since a DT of $n$ points can theoretically have up to O($n^2$) tetrahedra.

### 3.4 Deletion Algorithms

The deletion of a single vertex in a Delaunay tetrahedralization is often simply referred to as the 'inverse of the incremental insertion algorithm'. Like the insertion operation, it is a *local* operation that involves modifying only some adjacent tetrahedra of a DT. Figure 5 illustrates a two-dimensional example where the vertex $x$ is deleted from a Delaunay triangulation. Although the problem appears to be simple, it is a much more difficult task to implement than the insertion of a point, and very few algorithms can be found in the literature.

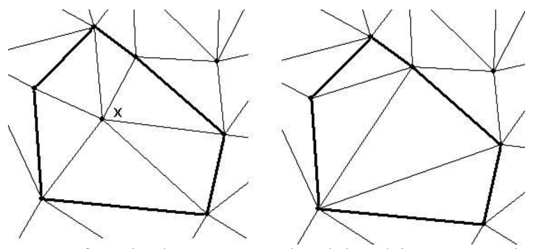

Figure 5. Left: $x$ is the vertex to be deleted in a 2D Delaunay triangulation. Right: re-triangulation of the polygon.

The most elegant algorithm, which is valid in any dimensions, is by Devillers (2002). In two dimensions, the method involves deleting all the triangles incident to the vertex $x$ and re-triangulating the hole by using a priority queue of the potential triangles that could be used to fill the hole. Devillers' algorithm states that the potential triangle having the smallest *power* – the power is a geometric function defined in Aurenhammer (1987) – with respect to $x$ is guaranteed to be Delaunay. Because the operation is local, the number of edges $k$ incident to a vertex is usually used to analyse deletion algorithms. Devillers' method has a time complexity of O($k$ log $k$) in two dimensions. Although possible, the implementation of the algorithm in 3D requires many modifications to handle the degenerate cases, and, to our knowledge, has not been implemented yet. Because the number of tetrahedra present in a Delaunay tetrahedralization of $n$ points varies depending on the configuration of the points, the time complexity of the method in 3D is O($t$ log $k$), where $t$ is the number of tetrahedra needed to fill the hole. A simpler solution involves keeping a list of all potential tetrahedra and testing (Delaunay empty *circumsphere* test) them against each vertex forming the hole. The resulting algorithm is slower – a time complexity of O($t$ $k$) – and an implementation can be found in CGAL (Devillers and Teillaud, 2003).

However, these methods temporarily destroy the tetrahedralization and some problems can arise. For this reason, we have developed a method that uses the flips described in Section 3.1 and an algorithm similar to the one implemented in CGAL. The methods works fine for most cases and we are currently working on making it fully robust for all the degenerate cases.

### 3.5 Movement of Points

When a point is continually moving over time, it makes no sense to continually insert, delete and re-insert it again somewhere else, because it is a costly operation. Instead, it can simply be moved and the topological relationships locally updated when it is needed. Roos (1991) and Gold (1991) detail a method that uses flipping to update the adjacency relationships of a 2D Delaunay triangulation as one vertex is moving over time. The movement of only one vertex to another location involves updating, by flips, all the topological relationships that will be modified from the starting point to the end point. If the location of the point is just slightly changed, the topological relationships will probably not need to be updated, but as soon as the moving point enters or leaves the circumcircle of a neighbouring triangle, a flip must be performed.

These ideas generalize to three dimensions, although, to our knowledge, no implementation is known. We are also currently working on implementing the method by using flip23 and flip32 to update the DT as one vertex is moving.

### 3.6 Possible Data Structures

When choosing a data structure to store a Delaunay tetrahedralization (or a Voronoi diagram), there is a trade-off between the size of the data structure and the topological relationships stored. For example, a very simple data structure means that when some operations are performed more work will have to be made (e.g. to retrieve the boundaries of Voronoi cells), while a data structure where the DT and its dual are both stored will speed up the use of these operations.

The simplest data structure to store the DT is the tetrahedron-based data structure where each record represents a tetrahedron with four pointers to its vertices and four pointers to its neighbouring tetrahedra. Many implementations of the DT (e.g. CGAL) use this data structure because it is simple and yields a fast construction. However, in our case, the VD will be needed for many spatial analysis functions and storing both might be advantageous. One solution is using the *facet-edge* structure (Dobkin and Laszlo, 1989), which stores symmetrically both the primal and dual of a three-dimensional subdivisions. As it name implies, the 'atom' of the structure is a pair of an edge and a face and operators to navigate from edges to edges on a same face or to visit all the faces incident to a given edge are available. Construction operators are also available. Although this solution seems attractive, it has been found difficult to implement in practice and, to our knowledge, has not been used for 'real projects'.

We are currently working on a simpler data structure, the 'augmented *quad-edge*', that also stores symmetrically the primal and dual 3D subdivisions. It uses the popular *quad-edge* structure (Guibas and Stolfi, 1985) originally developed for 2D subdivisions to store individually each cell (tetrahedron or Voronoi cell). The cells are linked together by the dual edge to the face shared by the two cells. The data structure is very simple to implement as only the *quad-edge* operators with minor modifications are needed to construct and modify the DT and the VD at the same time. The major limitation of this data structure is its high storage requirements.

## 4. VORONOI DIAGRAM AS A SPATIAL MODEL

Two-dimensional GIS's vector-based representations describe individually each object with geometric primitives, usually points, lines and polygons. The structure of these systems is based on the 'overlays', i.e. that the topology between objects is based on the intersection of lines, and, moreover, the building of this topology is a *global* process that needs to be done each time there is a modification on the map. The vectorial representation has also been extended to 3D, for example by Molenaar (1992), but the same 'problems' are present. Modelling three-dimensional oceanographic data with such a spatial model is obviously impossible because, firstly, the definition of topology is not appropriate, and secondly, the movement of objects is almost impossible.

The other spatial model used in the GIS and 3D modelling systems is the raster representation. Such a spatial model divides the space into regular cells that are usually squares in 2D and cubes in 3D (this is also called a *voxel* representation). The raster representation is particularly useful to represent fields or continuous phenomena because cells cover the whole space. Although being a popular representation in geosciences because the model gives a simple definition of spatial relationships, it cannot represent each object individually (the original data are 'lost' when converting them to raster) and the volume of data can become enormous if one wants to have a fine resolution, especially in 3D. To overcome the latter problem, the *octree* (Samet, 1990), where voxels are indexed and merged together to save space, can be used.

A spatial model for oceanographic data should ideally be able to represent both discrete objects and continuous phenomena. The Voronoi diagram has properties from both the vector and the raster spatial models: each individual object can be represented, and the ‘tiling’ properties give a definition of adjacency even for unconnected objects (each point generates one cell and this cell has some neighbours). Field-type data can be represented by assigning an attribute value to each Voronoi cell. There are several reasons for using a spatial model based on the Voronoi diagram over other models:

1. This is an adaptive method, i.e. the size of the cells depends on the distribution of the data points.
2. This is an automatic method that does not need user-defined parameters to be constructed.
3. Original data are kept and not ‘lost’, as is the case with raster representation.
4. By using the dual of the VD, the Delaunay tetrahedralization, the rendering is optimised since triangular elements are the primitives of choice for most graphics packages and video cards.
5. Local updates to the model are possible.

## 5. 3D SPATIAL ANALYSIS FUNCTIONS AND APPLICATIONS

Once the VD/DT is built, many spatial analysis operations are possible and even simplified. This section gives some examples of these.

### 5.1 Spatial Interpolation with the VD

Interpolation methods are used to estimate the value of an attribute at unsampled locations. They are required to model, visualise and better understand a dataset, and also to convert a dataset to another format (e.g. from scattered data to voxels). Traditional GIS interpolation methods (e.g. distance-based and triangle-based methods) can be generalized to 3D but they have many flaws when dealing with datasets having a highly irregular distribution. These flaws are caused by the fact that these methods do not consider the configuration of the data. It has been shown that natural neighbour interpolation (Sibson, 1981) avoids most of the problem of traditional methods and performs well for irregularly distributed data (Gold, 1989; Watson, 1992). This is a method entirely based on the Voronoi diagram for both selecting the neighbours and assigning a weight to each of them; and it is valid in any dimensions. To interpolate at location $x$ in 3D, a temporary point $x$ must be inserted into the VD. The neighbours involved in the interpolation process are the neighbours of $x$, and the weight of each is defined by the volume that the Voronoi cell of $x$ steals from the Voronoi cell of the neighbour in the absence of $x$.

Although the method has been implemented with success for 2D applications (Watson, 1992), its use in 3D is quite limited because its implementation is a complicated process that requires the computation of two VD – one with $x$ and another one without – and also of the volume of Voronoi cells. An algorithm that uses flipping and an incremental insertion algorithm, as described in Sections 3.1 and 3.3, has recently been developed by the authors of this paper (Ledoux and Gold, 2004). The algorithm is efficient (its time complexity is the same as the one for inserting a single point in a VD/DT) and we believe it to be considerably simpler to implement than other known methods, as only an incremental algorithm based on flips, with some minor modifications, is needed.

### 5.2 Extraction of Iso-Surfaces

It is notorious that three-dimensional data are very difficult to visualise, even within a 3D environment that offers translation, rotation and zoom functions. One of the best ways to better understand a dataset is to extract and visualise in 3D an iso-surface from it. An iso-surface (see Figure 6), also called an implicit surface, is the three-dimensional analogous concept of an iso-contour line in two dimensions: it represents the space where the attribute of a dataset has the same value. The most common algorithm to extract iso-surfaces, called *marching cubes* (Lorensen and Cline, 1987), was designed to work with a voxel input only. This algorithm can nevertheless be easily rewritten to work with a set of adjacent tetrahedra instead of cubes: each tetrahedron of a DT is visited and the intersections between the implicit surface and each edge of the tetrahedron are computed by linear interpolation. There are three possible cases for each tetrahedron: no intersection; three edges intersect and therefore a triangular face of the implicit surface is created; or four intersections, in that case two triangular faces must be created. The resulting implicit surface is formed of many adjacent, but topologically unconnected, triangular faces, which is ideal for fast rendering.

With the new techniques developed in recent years in computer graphics, it is possible to draw many iso-surfaces and view them all by using 'transparency' techniques, by assigning different colours to each, by 'peeling off' surfaces and by navigating inside and outside to see the shape. Visualisation therefore plays an important role in understanding a dataset, as it becomes a form of qualitative spatial analysis. Head et al. (1997) give more examples of how visualisation can help to better understand an oceanographic dataset.

### 5.3 Temporal Data and Real-Time Applications

The VD permits insertion, deletion and movement of objects with local modifications only; thus every operation is reversible. As shown in Gold (1996), by simply keeping a 'log file' of every operation done it is possible to rebuild each topological state of a map, at any time. This solves a big problem with temporal data and GIS, and it is valid both in 2D and 3D. There is no need to keep various 'snapshots' on the data at different time for further analysis: when a map a at specific time is desired, it is reconstructed from the original data from the log file. A map can also be viewed like a 'movie' of the changes during a certain period of time with for example boats and water moving.

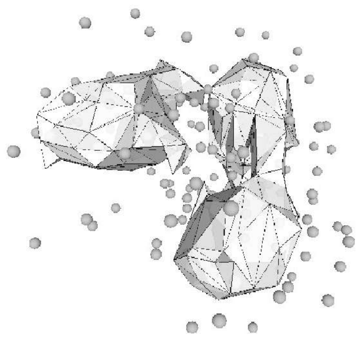

Figure 6. Example of an iso-surface extracted from 3D data points.

This spatial model also permits 'real-time' applications, i.e. as data are collected at sea, they can be quickly processed and added to the system for analysis, without rebuilding the whole topological relationships. This permits us to directly evaluate at sea the quality of a survey done and to correct mistakes or add new data while the boat is still near the site. Hatcher and Maher (1999) present more examples of real-time GIS applications at sea.

## 6. DISCUSSION

The main objective of our research is to build a complete spatial model to manage and analyse oceanographic data. We have presented the main properties of the 3D Voronoi diagram/Delaunay tetrahedralization and showed that it can indeed solve most of the problems arising when other methods are used. We have already implemented many construction and modification operators and we are planning to implement all the algorithms discussed in this paper. We have also developed some 3D spatial analysis functions and are currently working on building a more complete list. Finally, the results of this research are not only limited to oceanographic data, as the needs for modelling these data are similar to the needs in other fields, such as geology and atmospheric sciences.

## 7. ACKNOWLEDGEMENTS

The first author would like to thank the support from Hong Kong's Research Grants Council through a research studentship. The second author acknowledges the Research Grants Council, Hong Kong, project PolyU 5068/00E for the support of this research.

## 8. REFERENCES

Aurenhammer, F., 1987. Power diagrams: properties, algorithms and applications. *SIAM J. Comput.*, 16: 78-96.

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

Brown, K.Q., 1979. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5): 223-228.

Cignoni, P., Montani, C. and Scopigno, R., 1998. DeWall: a fast divide & conquer Delaunay triangulation algorithm in $E^d$. *Computer-Aided Design*, 30(5): 333-341.

Davis, B.E. and Davis, P.E., 1988. Marine GIS: Concepts and Considerations, Proc. GIS/LIS '88, Falls Church, VA, USA.

Devillers, O., 2002. On Deletion in Delaunay Triangulations. International *Journal of Computational Geometry and Applications*, 12(3): 193-205.

Devillers, O. and Teillaud, M., 2003. Perturbations and Vertex Removal in a 3D Delaunay Triangulation, *Proc. 14th ACM-SIAM Symp. Discrete Algorithms (SODA)*, Baltimore, MD, USA, pp. 313-319.

Dobkin, D.P. and Laszlo, M.J., 1989. Primitives for the Manipulation of Three-Dimensional Subdivisions. *Algorithmica*, 4: 3-32.

Field, D.A., 1986. Implementing Watson's algorithm in three dimensions, *Proc. 2nd Annual Symp. Computational Geometry*. ACM Press, Yorktown Heights, New York, USA.

Gold, C.M., 1989. Surface Interpolation, spatial adjacency and GIS. In: J. Raper (Ed.), *Three Dimensional Applications in Geographic Information Systems*. Taylor & Francis, pp. 21-35.

Gold, C.M., 1991. Problems with Handling Spatial Data – the Voronoi Approach. *CISM Journal*, 45(1): 65-80.

Gold, C.M., 1996. An Event-Driven Approach to Spatio-Temporal Mapping. *Geomatica, Journal of the Canadian Institute of Geomatics*, 50(4): 415-424.

Gold, C.M., Charters, T.D. and Ramsden, J., 1977. Automated contour mapping using triangular element data structures and an interpolant over each triangular domain. In: J. George (Editor), *Proc. Siggraph '77*. Computer Graphics, pp. 170-175.

Gold, C.M. and Condal, A.R., 1995. A Spatial Data Structure Integrating GIS and Simulation in a Marine Environment. *Marine Geodesy*, 18: 213-228.

Green, P.J. and Sibson, R., 1978. Computing Dirichlet tessellations in the plane. *The Computer Journal*, 21(2): 168-173.

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: 74-123.

Hatcher, G.A.J. and Maher, N., 1999. Real-time GIS for Marine Applications. In: D.J. Wright and D. Bartlett (Eds), *Marine and Coastal Geographic Information Systems*. Taylor & Francis, London, pp. 137-147.

Head, M.E.M., Luong, P., Costolo, J.H., Countryman, K. and Szczechowski, C., 1997. Applications of 3-D visualizations of oceanographic data bases, *Proc. Oceans '97-MTS/IEEE*, pp. 1210-1215.

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

Jones, C.B., 1989. Data structures for three-dimensional spatial information systems in geology. *International Journal of Geographic Information Systems*, 3(1): 15-31.

Lorensen, W.E. and Cline, H.E., 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 4: 163-168.

Ledoux, H. and Gold, C.M., 2004. An Efficient Natural Neighbour Interpolation Algorithm for Geoscientific Modelling, *Proc. 11th Int. Symp. Spatial Data Handling* (23-25 August 2004), Leicester, UK. *to appear.*

Li, R. and Saxena, N.K., 1993. Development of an Integrated Marine Geographic Information System. *Marine Geodesy*, 16: 293-307.

Lockwood, M. and Li, R., 1995. Marine Geographic Information Systems – What Sets Them Apart? *Marine Geodesy*, 18: 157-159.

Molenaar, M., 1992. A topology for 3D vector maps. *ITC Journal*, 1: 25-33.

Mücke, E.P., Saias, I. and Zhu, B., 1999. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. *Computational Geometry*, 12: 63-83.

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

Raper, J., (Ed.) 1989. *Three Dimensional Applications in Geographic Information Systems*. Taylor & Francis, London.

Roos, T., 1991. Dynamic Voronoi diagrams, Universität Würzburg, Germany.

Samet, H., 1990. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 493 pp.

Sibson, R., 1981. A brief description of natural neighbour interpolation. In: V. Barnett (Editor), *Interpreting Multivariate Data*. Wiley, New York, USA, pp. 21-36.

Watson, D.F., 1981. Computing the *n*-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2): 167-172.

Watson, D.F., 1992. *Contouring: A Guide to the Analysis and Display of Spatial Data*. Pergamon Press, Oxford, UK.

Wright, D.J. and Goodchild, M.F., 1997. Data from the Deep: Implications for the GIS Community. *The International Journal of Geographical Information Science*, 11(6): 523-528.