# An Efficient Natural Neighbour Interpolation Algorithm for Geoscientific Modelling*

Hugo Ledoux and Christopher Gold

Department of Land Surveying and Geo-Informatics
The Hong Kong Polytechnic University, Hong Kong
hugo.ledoux@polyu.edu.hk — christophergold@voronoi.com

## Abstract

Although the properties of natural neighbour interpolation and its usefulness with scattered and irregularly spaced data are well-known, its implementation is still a problem in practice, especially in three and higher dimensions. We present in this paper an algorithm to implement the method in two and three dimensions, but it can be generalized to higher dimensions. Our algorithm, which uses the concept of flipping in a triangulation, has the same time complexity as the insertion of a single point in a Voronoi diagram or a Delaunay triangulation.

## 1 Introduction

Datasets collected to study the Earth usually come in the form of two- or three-dimensional scattered points to which attributes are attached. Unlike datasets from fields such as mechanical engineering or medicine, geoscientific data often have a highly irregular distribution. For example, bathymetric data are collected at a high sampling rate along each ship's track, but there can be a very long distance between two ships' tracks. Also, geologic and oceanographic data respectively are gathered from boreholes and water columns; data are therefore usually abundant vertically but sparse horizontally. In order to model, visualize and better understand these datasets, interpolation is performed to estimate the value of an attribute at unsampled locations. The abnormal distribution of a dataset causes many problems for interpolation methods, especially for traditional weighted average methods in which distances are used to select neighbours and to assign weights. Such methods have problems because they do not consider the configuration of the data.

*This research is supported by the Hong Kong's Research Grants Council (project PolyU 5068/00E).

It has been shown that natural neighbour interpolation (Sibson, 1980, 1981) avoids most of the problems of conventional methods and therefore performs well for irregularly distributed data (Gold, 1989; Sambridge et al., 1995; Watson and Phillip, 1987). This is a weighted average technique based on the Voronoi diagram (VD) for both selecting the set of neighbours of the interpolation point $x$ and determining the weight of each. The neighbours used in an estimation are selected using the adjacency relationships of the VD, which results in the selection of neighbours that both surround and are close to $x$. The weight of each neighbour is based on the volume (throughout this paper, 'volume' is used to define area in 2D, volume in 3D and hyper volume in higher dimensions) that the Voronoi cell of $x$ 'steals' from the Voronoi cell of the neighbours in the absence of $x$. The method, which has many useful properties valid in any dimensions, is further defined in Sect. 2.

Although the concepts behind natural neighbour interpolation are simple and easy to understand, its implementation is far from being straightforward, especially in higher dimensions. The main reasons are that the method requires the computation of two Voronoi diagrams—one with and one without the interpolation point—and also the computation of volumes of Voronoi cells. This involves algorithms for both constructing a VD—or its geometric dual the Delaunay triangulation (DT)—and deleting a point from it. By comparison, conventional interpolation methods are relatively easy to implement; this is probably why they can be found in most geographical information systems (GIS) and geoscientific modelling packages.

Surprisingly, although many authors present the properties and advantages of the method, few discuss details concerning its implementation. The two-dimensional case is relatively easy to implement as efficient algorithms for constructing a VD/DT (Fortune, 1987; Guibas and Stolfi, 1985; Watson, 1981) and deleting a point from it (Devillers, 2002; Mostafavi et al., 2003) exist. Watson (1992) also presents an algorithm that mimics the insertion of $x$, and thus deletion algorithms are not required. The stolen area is obtained by ordering the natural neighbours around $x$ and decomposing the area into triangles.

In three and higher dimensions, things get more complicated because the algorithms for constructing and modifying a VD/DT are still not well-known. There exist algorithms to construct a VD/DT (Edelsbrunner and Shah, 1996; Watson, 1981), but deletion algorithms are still a problem—only theoretical solutions exist (Devillers, 2002; Shewchuk, 2000). Sambridge et al. (1995) describe three-dimensional methods to compute a VD, insert a new point in it and compute volumes of Voronoi polyhedra, but they do not explain how the interpolation point can be deleted. Owen (1992) also proposes a sub-optimal solution in which, before inserting the interpolation point $x$, he simply saves the portion of the DT that will be modified and replaces it once the estimation has been computed. The stolen volumes are calculated in only one operation, but that requires algorithms for intersecting planes in three-dimensional space. The idea of mimicking the insertion algorithm of

Watson (1992) has also been generalized to three dimensions by Boissonnat and Cazals (2002) and to arbitrary dimensions by Watson (2001). To calculate the stolen volumes, both algorithms use somewhat complicated methods to order the vertices surrounding $x$ and then decompose the volume into simplices (tetrahedra in three dimensions). The time complexity of these two algorithms is the same as the one to insert one point in a VD/DT.

We present in this paper a simple natural neighbour interpolation algorithm valid in two and three dimensions, but the method generalizes to higher dimensions. Our algorithm works directly on the Delaunay triangulation and uses the concept of *flipping* in a triangulation, as explained in Sect. 3, for both inserting new points in a DT and deleting them. The Voronoi cells are extracted from the DT and their volumes are calculated by decomposing them into simplices; we show in Sect. 4 how this step can be optimised. 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 insertion algorithm based on flips, with some minor modifications, is needed.

## 2 Natural Neighbour Interpolation

The idea of a natural neighbour is closely related to the concepts of the Voronoi diagram and the Delaunay triangulation. Let $S$ be a set of $n$ points in $d$-dimensional space. The Voronoi cell of a point $p \in S$, defined $\mathcal{V}_p$, is the set of points $x$ that are closer to $p$ than to any other point in $S$. The union of the Voronoi cells of all generating points $p$ in $S$ form the Voronoi diagram (VD) of $S$. The geometric dual of VD($S$), the Delaunay triangulation DT($S$), partitions the same space into simplices—a simplex represents the simplest element in a given space, e.g. a triangle in 2D and a tetrahedron in 3D—whose *circumspheres* do not contain any other points in $S$. The vertices of the simplices are the points generating each Voronoi cell. Fig. 1(a) shows the VD and the DT in 2D, and Fig. 1(b) a Voronoi cell in three dimensions. The VD and the DT represent the same thing: a DT can always be extracted from a VD, and vice-versa. The natural neighbours of a point $p$ are the points in $S$ sharing a Delaunay edge with $p$, or, in the dual, the ones whose Voronoi cell is contiguous to $\mathcal{V}_p$. For example, in Fig. 1(a), $p$ has seven natural neighbours.

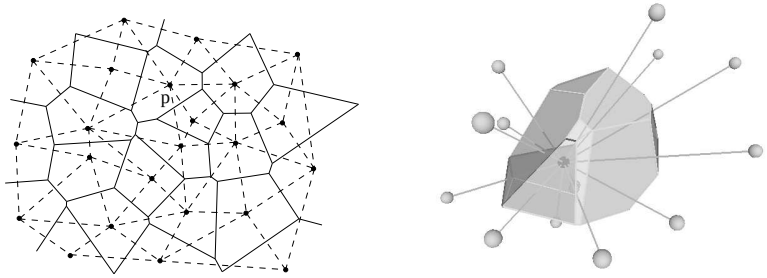
### 2.1 Natural Neighbour Coordinates

The concept of natural neighbours can also be applied to a point $x$ that is not present in $S$. In that case, the natural neighbours of $x$ are the points in $S$ whose Voronoi cell would be modified if the point $x$ were inserted in VD($S$). The insertion of $x$ creates a new Voronoi cell $\mathcal{V}_x$ that 'steals' volume from the Voronoi cells of its 'would be' natural neighbours, as shown in Fig. 2(a). This idea forms the basis of natural neighbour coordinates (Sibson, 1980, 1981),

which define quantitatively the amount $\mathcal{V}_x$ steals from each of its natural neighbours. Let $\mathcal{D}$ be the VD($S$), and $\mathcal{D}^+ = \mathcal{D} \cup \{x\}$. The Voronoi cell of a point $p$ in $\mathcal{D}$ is defined by $\mathcal{V}_p$, and $\mathcal{V}_p^+$ is its cell in $\mathcal{D}^+$. The natural neighbour coordinate of $x$ with respect to a point $p_i$ is

$$w_i(x) = \frac{Vol(\mathcal{V}_{p_i} \cap \mathcal{V}_x^+)}{Vol(\mathcal{V}_x^+)} \tag{1}$$

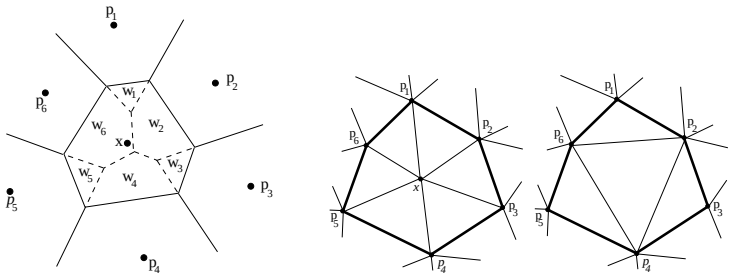
where $Vol(\mathcal{V}_{p_i})$ represents the volume of $\mathcal{V}_{p_i}$. For any $x$, the value of $w_i(x)$ will always be between 0 and 1: 0 when $p_i$ is not a natural neighbour of $x$, and 1 when $x$ is exactly at the same location as $p_i$. A further important consideration is that the sum of the volumes stolen from each of the $k$ natural neighbours is equal to $Vol(V_x^+)$. Therefore, the higher the value of $w_i(x)$ is, the stronger is the 'influence' of $p_i$ on $x$. The natural neighbour coordinates are influenced by both the distance from $x$ to $p_i$ and the spatial distribution of the $p_i$ around $x$.




(a) Voronoi diagram and Delaunay triangulation (dashed lines) in 2D.

(b) A Voronoi cell in 3D with its dual Delaunay edges joining the generator to its natural neighbours.

**Fig. 1.** Voronoi diagram.




(a) Natural neighbour coordinates of $x$ in 2D.

(b) 2D DT with and without $x$.

**Fig. 2.** Two VD are required for the natural neighbour interpolation.

### 2.2 Natural Neighbour Interpolation

Based on the natural neighbour coordinates, Robin Sibson developed a weighted average interpolation technique that he named natural neighbour interpolation (Sibson, 1980, 1981). The points used to estimate the value of an attribute at location $x$ are the natural neighbours of $x$, and the weight of each neighbour is equal to the natural neighbour coordinate of $x$ with respect to this neighbour. If we consider that each data point in $S$ has an attribute $a_i$ (a scalar value), the natural neighbour interpolation is

$$f(x) = \sum_{i=1}^{k} w_i(x)\, a_i \tag{2}$$

where $f(x)$ is the interpolated function value at the location $x$. The resulting method is exact ($f(x)$ honours each data point), and $f(x)$ is smooth and continuous everywhere except at the data points. To obtain a continuous function everywhere, that is a function whose derivative is not discontinuous at the data points, Sibson uses the weights defined in Eq. 1 in a quadratic equation where the gradient at $x$ is considered. To our knowledge, this method has not been used with success with real data and therefore we do not use it. Other ways to remove the discontinuities at the data points have been proposed: Watson (1992) explains different methods to estimate the gradient at $x$ and how to incorporate it in Eq. 2; and Gold (1989) proposes to modify the weight of each $p_i$ with a simple hermitian polynomial so that, as $x$ approaches $p_i$, the derivative of $f(x)$ approaches 0.

Modifying Eq. 2 to obtain a continuous function can yield very good results in some cases, but with some datasets the resulting surface can contain unwanted effects. Different datasets require different methods and parameters, and, for this reason, modifications should be applied with great care.

### 2.3 Comparison with Other Methods

With traditional weighted average interpolation methods, for example distance-based methods, all the neighbours within a certain distance from the interpolation location $x$ are considered and the weight of each neighbour is inversely proportional to its distance to $x$. These methods can be used with a certain success when the data are uniformly distributed, but it is difficult to obtain a continuous surface when the distribution of the data is anisotropic or when there is variation in the data density. Finding the appropriate distance to select neighbours is difficult and requires *a priori* knowledge of a dataset. Natural neighbour interpolation, by contrast, is not affected by these issues because the selection of the neighbours is based on the configuration of the data.

Another popular interpolation method, especially in the GIS community, is the triangle-based method in which the estimate is obtained by linear interpo-

lation within each triangle, assuming a triangulation of the data points is available. The generalization of this method to higher dimensions is straightforward: linear interpolation is performed within each simplex of a $d$-dimensional triangulation. In 2D, when this method is used with a Delaunay triangulation, satisfactory results can be obtained because the Delaunay criterion maximizes the minimum angle of each triangle, i.e. it creates triangles that are as equilateral as possible. This method however creates discontinuities in the surface along the triangle edges and, if there is anisotropy in the data distribution, the three neighbours selected will not necessarily be the three closest data points. These problems are amplified in higher dimensions because, for example, the *max-min* angle property of a DT does not generalize to three dimensions. A 3D DT can contain some tetrahedra, called *slivers*, whose four vertices are almost coplanar; interpolation within such tetrahedra does not yield good results. The presence of slivers in a DT does not however affect natural neighbour interpolation because the Voronoi cells of points forming a sliver will still be 'well-shaped' (relatively spherical).

## 3 Delaunay Triangulation, Duality and Flips

In order to construct and modify a Voronoi diagram, it is actually easier to first construct the Delaunay triangulation and extract the VD afterwards. Managing only simplices is simpler then managing arbitrary polytopes: the number of vertices and neighbours of each simplex is known and constant which facilitates the algorithms and simplifies the data structures. Extracting the VD from a DT in 2D is straightforward, while in 3D it requires more work. In two dimensions the dual of a triangle is a point (the centre of the circumcircle of the triangle) and the dual of a Delaunay edge is a bisector edge. In three dimensions the dual of a tetrahedron is a point (the centre of the circumsphere of the tetrahedron) and the dual of a Delaunay edge is a Voronoi face (a convex polygon formed by the centre of the circumspheres of every tetrahedron incident to the edge). In short, to get the Voronoi cell of a given point $p$ in a 3D DT, we must first identify all the edges that have $p$ as a vertex and then extract the dual of each (a face). The result will be a convex polyhedron formed by convex faces, as shown in Fig. 1(b).

We discuss in this section the main operations required for the construction of a DT and for implementing the natural neighbour interpolation algorithm described in Sect. 4. Among all the possible algorithms to construct a VD/DT, we chose an incremental insertion algorithm because it permits to firstly construct a DT and then modify it locally when a point is added or deleted. Other potential solutions, for example divide-and-conquer algorithms or the construction of the convex hull in $(d+1)$ dimensions, might be useful for the initial construction, but local modifications are either slow and complicated, or simply impossible.

### 3.1 Flipping

A flip is a local topological operation that modifies the configuration of adjacent simplices in a triangulation. Consider the set $S = \{a, b, c, d\}$ of points in the plane forming a quadrilateral, as shown in Fig. 3(a). There exist exactly

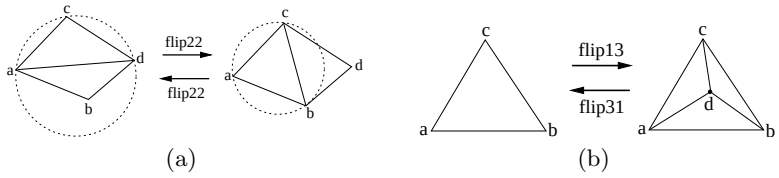



**Fig. 3.** Two-dimensional flips.

two ways to triangulate $S$: the first one contains the triangles $abc$ and $bcd$; and the second one contains the triangles $abd$ and $acd$. Only the first triangulation of $S$ is Delaunay because $d$ is outside the circumcircle of $abc$. A *flip22* is the operation that transforms the first triangulation into the second, or vice-versa. It should be noticed that when $S$ does not form a quadrilateral, as shown in Fig. 3(b), there is only one way to triangulate $S$: with three triangles all incident to $d$. A *flip13* refers to the operation of inserting $d$ inside the triangle $abc$ and splitting it into three triangles; and a *flip31* is the inverse operation that is needed for deleting $d$. The notation for the flips refers to the numbers of simplices before and after the flip.

The concept of flipping generalizes to three and higher dimensions (Lawson, 1986). The flips to insert and delete a point generalize easily to three dimensions and become respectively *flip14* and *flip41*, as shown in Fig. 4(b). The generalization of the flip22 in three dimensions is somewhat more complicated. Consider a set $S = \{a, b, c, d, e\}$ of points in $\mathbb{R}^3$, as shown in Fig. 4(a). There are two ways to triangulate $S$: either with two or three tetrahedra. In the first case, the two tetrahedra share a face, and in the latter case the three

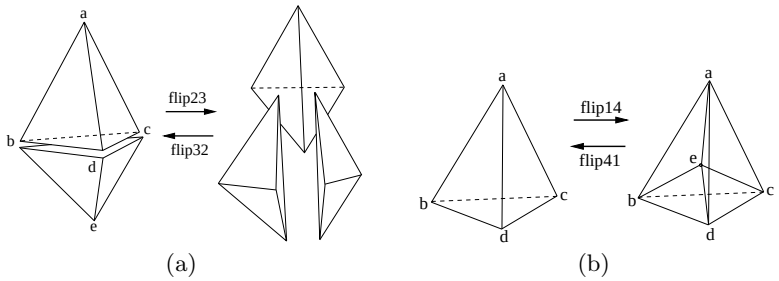



**Fig. 4.** Three-dimensional flips.

tetrahedra all have a common edge. A *flip23* transforms a triangulation of two tetrahedra into another one containing three tetrahedra; a *flip32* is the inverse operation.

### 3.2 Constructing a DT by Flips

Consider a $d$-dimensional Delaunay triangulation $\mathcal{T}$ and a point $x$. What follow are the steps to insert $x$ in $\mathcal{T}$ by flips, assuming that the simplex $\tau$ containing $x$ has been identified (see Devillers et al. (2002) for different methods). After the insertion of $x$, one or more simplices of $\mathcal{T}$ will be in 'conflict' with $x$, i.e. their circumspheres will contain $x$. We must identify, delete and replace these conflicting simplices by other ones. A flipping algorithm first splits $\tau$ into $d+1$ simplices with a flip (e.g. a flip13 in 2D). Each new simplex must then be tested to make sure it is Delaunay; this test involves only two simplices: the new simplex and its adjacent neighbour that is not incident to $x$ (there is only one). If the new simplex is not Delaunay then a flip is performed. The new simplices thus created must be tested later. The process continues until every simplex incident to $x$ is Delaunay. This idea can be applied to construct a DT: each point is inserted one at a time and the triangulation is updated between each insertion.

This incremental insertion algorithm is valid in any dimensions, i.e. there always exists a sequence of flips that will permit the insertion of a single point in a $d$-dimensional DT. For a detailed description of the algorithm, see Guibas and Stolfi (1985) and Edelsbrunner and Shah (1996) for respectively the two- and $d$-dimensional case.

## 4 A Flip-Based Natural Neighbour Interpolation Algorithm

Our algorithm to implement natural neighbour interpolation performs all the operations directly on the Delaunay triangulation (with flips) and the Voronoi cells are extracted when needed. We use a very simple idea that consists of inserting the interpolation point $x$ in the DT, calculating the volume of the Voronoi cell of each natural neighbour of $x$, then removing $x$ and recalculating the volumes to obtain the stolen volumes. Two modifications are applied to speed up the algorithm. The first one concerns the deletion of $x$ from the DT. We show in Sect. 3 that every flip has an 'inverse', e.g. in 2D, a flip13 followed by a flip31 does not change the triangulation; in 3D, a flip23 creates a new tetrahedron that can then be removed with a flip32. Therefore, if $x$ was added to the triangulation with a sequence $l$ of flips, simply performing the inverse flips of $l$ in reverse order will delete $x$. The second modification concerns how the overlap between Voronoi cells with and without the presence of $x$ is calculated. We show that only some faces of a Voronoi cell (in the following, a Voronoi face is a $(d-1)$-face forming the boundary of the cell, e.g. in 2D

it is a line and in 3D it is a polygon) are needed to obtain the overlapping volume.

Given a set of points $S$ in $d$ dimensions, consider interpolating at the location $x$. Let $\mathcal{T}$ be the DT($S$) and $p_i$ the natural neighbours of $x$ once it is inserted in DT($S$). The simplex $\tau$ that contains $x$ is known. Our algorithm proceeds as follow:

1. $x$ is inserted in $\mathcal{T}$, thus getting $\mathcal{T}^+ = \mathcal{T} \cup \{x\}$, by using flips and the sequence $l$ of flips performed is stored in a simple list.
2. the volume of $\mathcal{V}_x^+$ is calculated, as well as the volumes of each $\mathcal{V}_{p_i}^+$.
3. $l$ is performed in reverse order and the inverse flip is performed each time. This deletes $x$ from $\mathcal{T}^+$.
4. the volume of $\mathcal{V}_{p_i}$ are re-calculated to obtain the natural neighbour coordinates of $x$ with respect to all the $p_i$; and Eq. 2 is finally calculated.

To remember the order of flips in two dimensions, a simple list containing the order in which the $p_i$ became natural neighbours of $x$ is kept. The flip13 adds three $p_i$, and each subsequent flip22 adds one new $p_i$. In 3D, only a flip23 adds a new $p_i$ to $x$; a flip32 only changes the configuration of the tetrahedra around $x$. We store a list of edges that will be used to identify what flip was performed during the insertion of $x$. In the case of a flip23, we simply store the edge $xp_i$ that is created by the flip. A flip32 deletes a tetrahedron and modifies the configuration of the two others such that, after the flip, they are both incident to $x$ and share a common face $abx$. We store the edge $ab$ of this face. Therefore, in two dimensions, to delete $x$ we take one $p_i$, find the two triangles incident to the edge $xp_i$ and perform a flip22. When $x$ has only three natural neighbours, a flip31 deletes $x$ completely from the $\mathcal{T}^+$. In 3D, if the current edge is $xp_i$, a flip32 is used on the three tetrahedra incident to $xp_i$; and if $ab$ is the current edge, then a flip23 is performed on the two tetrahedra sharing the face $abx$.

### 4.1 Volume of a Voronoi Cell

The volume of a $d$-dimensional Voronoi cell is computed by decomposing it into $d$-simplices and summing their volumes. The volume of a $d$-simplex $\tau$ is easily computed:

$$Vol(\tau) = \frac{1}{d!} \left| det \begin{pmatrix} v^0 & \dots & v^d \\ 1 & \dots & 1 \end{pmatrix} \right| \tag{3}$$

where $v^i$ is a $d$-dimensional vector representing the coordinates of a vertex and $det$ is the determinant of the matrix. Triangulating a 2D Voronoi cell is easily performed: since the polygon is convex a fan-shaped triangulation can be done. In 3D, the polyhedron is triangulated by first fan-shaped triangulating each of its Voronoi faces, and then the tetrahedra are formed by the triangles and the generator of the cell.

In order to implement natural neighbour interpolation, we do not need to know the volume of the Voronoi cells of the $p_i$ in $\mathcal{T}$ and $\mathcal{T}^+$, but only the

*difference* between the two volumes. As shown in Fig. 2(a), some parts of a Voronoi cell will not be affected by the insertion of $x$ in $\mathcal{T}$, and computing them twice to subtract afterwards is computationally expensive and useless. Notice that the insertion or deletion of $x$ in a DT modifies only locally the triangulation—only simplices inside a defined polytope (defined by the $p_i$ in Fig. 2(b)) are modified. Each $p_i$ has many edges incident to it, but only the edges inside the polytope are modified. Therefore, to optimise this step of the algorithm, we process only the Voronoi faces that are dual to the Delaunay edges joining two natural neighbours of $x$. In $\mathcal{T}^+$, the Voronoi face dual to the edge $xp_i$ must also be computed. Only the complete volume of the Voronoi cell of $x$ in $\mathcal{T}^+$ needs to be known.

The Voronoi cells of the points in $S$ forming the convex hull of $S$ are unbounded. That causes problems when a natural neighbour of $x$ is one of these points because the volume of its Voronoi cell, or parts of it, must be computed. The simplest solution consists of bounding $S$ with an artificial $(d+1)$-simplex big enough to contain all the points.

### 4.2 Theoretical Performances

By using a flipping algorithm to insert $x$ in a $d$-dimensional DT $\mathcal{T}$, each flip performed removes one and only one conflicting simplex from $\mathcal{T}$. For example, in 3D, the first flip14 deletes the tetrahedron containing $x$ and adds four new tetrahedra to $\mathcal{T}^+$; then each subsequent flip23 or flip32 deletes only one tetrahedron that was present in $\mathcal{T}$ before the insertion of $x$. Once a simplex is deleted after a flip, it is never re-introduced in $\mathcal{T}^+$. The work needed to insert $x$ in $\mathcal{T}$ is therefore proportional to $r$, the number of simplices in $\mathcal{T}$ that conflict with $x$. As already mentioned, each 2D flip adds a new natural neighbour to $x$. The number of flips needed to insert $x$ is therefore proportional to the degree of $x$ (the number of incident edges) after its insertion. Without any assumptions on the distribution of the data, the average degree of a vertex in a 2D DT is 6; which means an average of four flips are needed to insert $x$ (one flip13 plus three flip22). This is not the case in 3D (a flip32 does not add a new natural neighbour to $x$) and it is therefore more complicated to give a value to $r$. We can nevertheless affirm that the value of $r$ will be somewhere between the number of edges and the number of tetrahedra incident to $x$ in $\mathcal{T}^+$; these two values are respectively around 15.5 and 27.1 when points are distributed according to a Poisson distribution (Okabe et al., 1992). Because a flip involves a predefined number of adjacent simplices, we assume it is performed in constant time. As a result, if $x$ conflicts with $r$ simplices in $\mathcal{T}$ then $\mathrm{O}(r)$ time is needed to insert it.

Deleting $x$ from $\mathcal{T}^+$ also requires $r$ flips; but this step is done even faster than the insertion because operations to test if a simplex is Delaunay are not needed, nor are tests to determine what flip to perform. The volume of each Voronoi cell is computed only partly, and this operation is assumed to be done in constant time. In the natural neighbour interpolation algorithm, if $k$ is the

degree of $x$ in a $d$-dimensional DT, then the volume of $k$ Voronoi cells must be partly computed twice: with and without $x$ in $\mathcal{T}$. As a conclusion, our natural neighbour interpolation algorithm has a time complexity of $O(r)$, which is the same as an algorithm to insert a single point in a Delaunay triangulation. However, the algorithm is obviously slower by a certain factor since $x$ must be deleted and parts of the volumes of the Voronoi cells of its natural neighbours must be computed.

## 5 Conclusions

Many new technologies to collect information about the Earth have been developed in recent years and, as a result, more data are available. These data are usually referenced in two- and three-dimensional space, but so-called four-dimensional datasets—that is three spatial dimensions plus a time dimension—are also collected. The GIS, with its powerful integration and spatial analysis tools, seems the perfect platform to manage these data. It started thirty years ago as a static mapping tool, has recently evolved to three dimensions (Raper, 1989) and is slowly evolving to higher dimensions (Mason et al., 1994; Raper, 2000). Interpolation is an important operation in a GIS. It is crucial in the visualisation process (generation of surfaces or contours), for the conversion of data from one format to another, to identify bad samples in a dataset or simply to have a better understanding of a dataset. Traditional interpolation methods, although relatively easy to implement, do not yield good results, especially when used with datasets having a highly irregular distribution. In two dimensions, these methods have shortcomings that create discontinuities in the surface and these shortcomings are amplified in higher dimensions.

The method detailed in this paper, natural neighbour interpolation, although more complicated to implement, performs well with irregularly distributed data and is valid in any dimensions. We have presented a simple, yet efficient, algorithm that is valid in two, three and higher dimensions. We say 'simple' because only an incremental algorithm based on flips, with the minor modifications described, is required to implement our algorithm. We have already implemented the algorithm in two and three dimensions and we hope our method will make it possible for the GIS community to take advantage of natural neighbour interpolation for modelling geoscientific data.

## References

Boissonnat JD, Cazals F (2002) Smooth surface reconstruction via natural neighbour interpolation of distance functions. Computational Geometry, 22:185–203.

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

Devillers O, Pion S, Teillaud M (2002) Walking in a triangulation. International Journal of Foundations of Computer Science, 13(2):181–199.

Edelsbrunner H, Shah N (1996) Incremental Topological Flipping Works for Regular Triangulations. Algorithmica, 15:223–241.

Fortune S (1987) A Sweepline algorithm for Voronoi diagrams. Algorithmica, 2:153–174.

Gold CM (1989) Surface Interpolation, spatial adjacency and GIS. In J Raper, editor, Three Dimensional Applications in Geographic Information Systems, pages 21–35. Taylor & Francis.

Guibas LJ, Stolfi J (1985) Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. ACM Transactions on Graphics, 4:74–123.

Lawson CL (1986) Properties of $n$-dimensional triangulations. Computer Aided Geometric Design, 3:231–246.

Mason NC, O'Conaill MA, Bell SBM (1994) Handling four-dimensional georeferenced data in environmental GIS. International Journal of Geographic Information Systems, 8(2):191–215.

Mostafavi MA, Gold CM, Dakowicz M (2003) Delete and insert operations in Voronoi/Delaunay methods and applications. Computers & Geosciences, 29(4):523–530.

Okabe A, Boots B, Sugihara K, Chiu SN (1992) Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley and Sons.

Owen SJ (1992) An Implementation of Natural Neighbor Interpolation in Three Dimensions. Master's thesis, Brigham Young University, Provo, UT, USA.

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

Raper J (2000) Multidimensional Geographic Information Science. Taylor & Francis.

Sambridge M, Braun J, McQueen H (1995) Geophysical parameterization and interpolation of irregular data using natural neighbours. Geophysical Journal International, 122:837–857.

Shewchuk JR (2000) Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. In Proc. 16th Annual Symp. Computational Geometry, pages 350–359. ACM Press, Hong Kong.

Sibson R (1980) A vector identity for the Dirichlet tesselation. In Mathematical Proceedings of the Cambridge Philosophical Society, 87, pages 151–155.

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

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

Watson DF (1992) Contouring: A Guide to the Analysis and Display of Spatial Data. Pergamon Press, Oxford, UK.

Watson DF (2001) Compound Signed Decomposition, The Core of Natural Neighbor Interpolation in $n$-Dimensional Space. `http://www.iamg.org/naturalneighbour.html`.

Watson DF, Phillip G (1987) Neighborhood-Based Interpolation. Geobyte, 2(2):12–16.