GEOMETRICAL DEGENERACY REMOVAL BY VIRTUAL DISTURBANCES
An Application to Surface Reconstruction from Point Slice Samples

Oscar Ruiz
CAD CAM CAE Laboratory, EAFIT University, Medellin, COLOMBIA
oruiz@eafit.edu.co

Eliana Vasquez, Sebastian Peña, Miguel Granados
Erasmus Universitaet, Nederlands, Fraunhofer Inst. Comp. Graphics, Germany
Max Planck Inst. Informatiks, Germany
e.vasquezosorio@erasmusmc.nl, sebastian.pena.serna@igd.fraunhofer.de, granados@mpi-sb.mpg.de

Keywords: geometric degeneracy, voronoi diagram, delaunay triangulation, surface reconstruction, slice point sample.

Abstract: In surface reconstruction from slice samples (typical in medical imaging, coordinate measurement machines, stereolithography, etc.) the available methods attack the geometrical and topological aspects or a combination of these. Topological methods classify the events occurred in the 2-manifold between two consecutive slices. Geometrical methods synthesize the surface based on local proximity of contours in consecutive slices. Many of these methods work with modifications of Voronoi - Delaunay (VD) techniques, applied on slices $i$ and $i+1$. Superimposed 2D Voronoi Diagrams $VD_i$ and $VD_{i+1}$ (used in surface reconstruction) present topological problems if, for example, a site of $VD_i$ lies on an site or an edge of $VD_{i+1}$. The usual treatment of this problem in literature is to apply a geometrical disturbance to either $VD_i$ or $VD_{i+1}$, thus eliminating the degeneracy. Recent works seek to quantify the amount of the disturbance applied in relation to the probability distribution of the event “change in the topology of $VD$”. In this article, in contrast, virtual disturbances are proposed and implemented, which allow for the application of subsequent steps of the algorithm at hand (in this case, tetrahedra construction for surface reconstruction) regardless of to the geometrical exception. Tetrahedra (or any other downstream constructs) can then be instantiated as per non-degenerate conditions. Although this method is applied for surface reconstruction, it gives insight as to how to circumvent degeneracies in procedures based on VD methods.

1 INTRODUCTION

Degenerate conditions in geometric algorithms have been dealt with different ways: (i) by stating the same problem in different spaces with better conditioning, (ii) by increasing the real computation precision, (iii) by relying on rational numbers, with no rounding errors, and (iv) by disturbing the input for the geometrical algorithms, while at the same time estimating the probability of respecting the original problem topology. Strategies (i) and (ii) have been extensively applied in Numerical Analysis, for example, by generating equivalent linear systems with better manipulation properties. Alternative (iii) has been investigated, for example in (Computational Geometry Algorithm Library (Burnikel et al., 1999)), with exact computation paradigms. Strategy (iv) has given probability bounds for alteration of Voronoi-Delaunay topology upon numerical disturbance of degenerate events (see (Funke et al., 2005)).

Virtual Perturbations have been used in other contexts (see (Edelsbrunner and Mücke, 1990) for previous reports). In the strategy presented here, the problem at hand is analyzed, and the topological structure of a correct result is created in the form of objects, without immediate instantiation. This strategy assumes the possibility of detecting the degenerate condition. Beyond this point, no numerical manipulation is introduced. Instead, the generic objects are instantiated with the numerical information, and the algorithm proceeds. It should be noticed that none of the mentioned strategies solves the degeneracy problem. Each is suited for a particular domain of problems. The one presented here is clearly convenient when there is a finite number of topological configurations, that can be enumerated and distinguished.

The particular context in which this strategy is presented is the general problem of surface recon-
struction, from planar samples. Particular steps of the Boissonat & Geiger algorithm ((Boissonnat, 1988; Geiger, 1993)) have been changed in order to make them more robust (see also [(Ruiz et al., 2002; Ruiz et al., 2005)]). Section 2 gives the application context of the present work and reviews related literature. Section 3 describes the methodology applied and the procedures followed. Section 4 gives an account of the results, and section 5 concludes the article.

2 CONTEXT AND LITERATURE REVIEW

The algorithm proposed and implemented by Boissonat & Geiger in (Boissonnat, 1988; Geiger, 1993) (called here B+G) builds tetrahedra filling the space between two consecutive sampling planes \( i \) and \( i + 1 \). B+G is a fairly fast and robust algorithm, originally presenting weaknesses that have been corrected by complementary works. The boolean union of such tetrahedra renders the solid object cut by the sampling planes. This strategy basically uses geometrical proximity between contours to infer the existence of a surface. For this reason, over-stretched surfaces may be generated, joining local portions of contours which are close, while the contours have little to do with each other in the global sense. This effect may be diminished by applying a 2D shape similarity (2DSS) algorithm (see (Ruiz et al., 2002; Ruiz et al., 2005)). On the other hand, the tetrahedra are built by projecting the Voronoi Diagram (VD) of the point set in level \( i \), \( \text{VD}_i \), onto \( \text{VD}_{i+1} \), or vice versa. A degeneracy condition for B+G occurs when a Voronoi site of \( \text{VD}_i \) exactly lies on either a site or an edge of \( \text{VD}_{i+1} \). Such a condition produces a non-manifold and self-intersection condition in the surface so built is the treatment of such exceptional situations by rearranging the data for a smooth functioning of B+G the goal of this work.

2.1 Brief Review of the B+G Method

The B+G method was developed by Jean-Daniel Boissonnat ((Boissonnat, 1988)) at INRIA and later improved by Bernhard Geiger ((Geiger, 1993)).

The B+G method processes each pair of adjacent levels, \( \text{level}_i \) and \( \text{level}_j \) and creates a flat-faced polyhedral surface that joins the contours of both levels. The process is based on geometric closeness, supported over two geometric structures: the 2D Delaunay Triangulation \( \text{DT} \) and the 2D Voronoi Diagram \( \text{VD} \).

The B+G method divides the interior of the contours in triangles by creating the Delaunay Triangulation of the contour vertices (figures 2 and 3). After some processing, the Voronoi Diagrams belonging to the levels are used to create a planar graph named the Joint Voronoi Diagram (figure 4). This graph states
how the triangles in the levels are linked, by translating then to tetrahedrons (figure 5). Finally the triangles of the tetrahedrons facing the exterior are taken as the reconstructed surface (figure 6).

3 IMPROVEMENTS ON THE B+G METHOD

The Polyhedral Surface Method is based on the B+G method. The B+G method reconstructs incomplete surfaces and presents non manifold situations. The improved version solves the incomplete surface problem (section 4) and the non manifold situations are minimized by taking into account special cases when creating the joint Voronoi Diagram (section 3.3). Also minor changes, in implementation and conceptualization, were performed to reach better results. In the following sections a description of these improvements is given.

3.1 Satisfaction of Conditions

Before the construction of the Joint Voronoi Diagram, some conditions must be fulfilled by the Delaunay Triangulation $DT$ on each level:

**Condition 0: Completeness of $DT$** The triangulation should include all the edges which form the set of contours $C_i$ in the level. Condition 0 is formally stated in 1.

\[ \forall \text{edge } e \in C_i : \exists DE_{ij} \text{ s.t. } e = DE_{ij} \]  

**Condition 1: Partition of $DT$ by contours** The classification of every triangle in the Delaunay Triangulation as internal or external with respect to the contours must be possible. Let the union of all the external triangles be called External Region $ER$, and the union of all the internal triangles the Internal Region $IR$. Then, condition 1 is formalized as called.

\[ \forall DT_{ijk} \in DT : (DT_{ijk} \subseteq IR) \lor (DT_{ijk} \subseteq ER) \]  

**Condition 2: Confinement of circumcenters** The circumcenter of every Delaunay triangle $DT_{ijk}$ must lie inside the region to which $DT_{ijk}$ belongs. In formal terms.

\[
\begin{cases} 
\text{if } DT_{ijk} \subseteq IR \Rightarrow \text{circumcenter}(DT_{ijk}) \in IR \\
\text{if } DT_{ijk} \subseteq ER \Rightarrow \text{circumcenter}(DT_{ijk}) \in ER 
\end{cases}
\]  

Notice that the satisfaction of condition 0 leads to the satisfaction of condition 1 and vice versa. Condition 0/1 and 2 are dependent with each other; condition 0/1 must be fulfilled before condition 2 is checked. This detail is not considered in the original method, and some of the presented algorithms fail because of this omission.

3.2 Lifting of Internal Points

The B+G method adds some points in the interior of the some contours in the levels (figures 7 and 8).

When re-sampling the resulting surface using the original planes, ghost lines will appear inside the original contours as shown in figure 9. These lines appear...
because the added points create an approximate medial axis, which allows to construct the surface as per figure 10. To avoid such lines, the points inserted inside the contours are orthogonally projected on a level between the processed levels before the surface is finished. The distance between the created level and the original one is small, to avoid geometric and topological degeneracies.

3.3 Special Cases in the Creation of the Joint Voronoi Diagram

The Joint Voronoi Diagram of two consecutive levels results from intersecting the orthogonal projections of their Voronoi Diagrams on a common plane. The Joint Voronoi Diagram is formed by three kinds of nodes: $T_1$, $T_2$ and $T_{12}$. The $T_i$ nodes correspond to the Voronoi vertices belonging to the Voronoi Diagram on level $i$, with $i = 1, 2$. The $T_{12}$ nodes correspond to the intersection of two Voronoi edges.

Every node in the graph corresponds to a tetrahedron, and the union of all these tetrahedrons form the 3D Delaunay Triangulation of the contour points $P$ on both levels $i$ and $j$. Because the tetrahedrons that are translated from the graph are Delaunay tetrahedrons, they satisfy the “empty-sphere” condition, namely is, the sphere that circumscribes the tetrahedron does not contain any other point in $P$ except its vertices.

Each tetrahedron is created with four Delaunay vertices. See figure 5 where a $T_1$, $T_2$ and two $T_{12}$ tetrahedrons are translated from the Joint Voronoi Diagram in figure 4.

The special cases are generated when more than four Delaunay vertices lie on the surface of an empty sphere. In the implementation of the B+G method, when this situation is found a perturbation is applied to vertices. This perturbation leads, sometimes, to undesired surfaces, like the one shown in figure 11. The conceptual consideration of the special cases does not leave the election of the tetrahedrons to a random perturbation, and improves the reconstructed surfaces (figure 12).

When a special case is created, the construction of the Joint Voronoi Diagram becomes ambiguous because there is more than one configuration of nodes (therefore, tetrahedrons) that could be generated. If nodes of different configurations are kept together at the same time, the graph becomes inconsistent, because the faces of the related tetrahedrons intersect with each other and the number of connections between the nodes exceed the limit. The problem is solved when a valid configuration of nodes, (defined as the set of nodes whose related tetrahedrons properly share faces), is found and it is inserted into the graph. These special cases are not considered in the original method.

As an upper bound, at most six Delaunay vertices may share the same empty sphere, because on each
level the limit of co-circular vertices is three and the graph generation involves just two levels.

### 3.3.1 Case 1: Voronoi Vertex vs. Voronoi Edge

This case is generated when five Delaunay vertices are co-spherical, leading to a Voronoi vertex belonging to level \( i \) being projected on a Voronoi edge belonging to level \( j \), or vice versa. An example is shown in figures 13 and 14.

![Figure 13](image1.png)

**Figure 13:** Voronoi vertex vs. Voronoi edge case: Case 1. Vertices \( DV_1, DV_2, DV_3, DV_B \) and \( DV_D \) are co-spherical.

![Figure 14](image2.png)

**Figure 14:** Voronoi vertex vs. Voronoi edge case: Case 2. Projected case.

Each pair of Voronoi edges that intersect each other, generate a \( T_{12} \) tetrahedron. In this case three intersections are found, \( VE_{12} \) vs. \( VE_{BD}, VE_{23} \) vs. \( VE_{BD} \) and \( VE_{31} \) vs. \( VE_{BD} \). The creation of all these \( T_{12} \) tetrahedrons is illegal, because their faces intersect and one face is shared by more than two tetrahedrons.

The ambiguity of the situation is essentially shown when creating the \( T_i \) tetrahedron related to \( VV_{123} \) in figures 13 and 14. In this case, two different Delaunay vertices are found at the same distance to \( VV_{123} \) and the distance is the minimum among all the points, so, any of them could be used as the apex. These two found Delaunay vertices are the ones related to the Voronoi edge on which \( VV_{123} \) is projected. In figure 14 they are \( DV_B \) and \( DV_D \). Because there are two alternatives to choose from, this situation allows two configurations as solutions.

The election of the apex for the \( T_i \) tetrahedron between these Delaunay vertices, states that the distance from the elected vertex to \( VV_{123} \) is virtually smaller than the distance from the non-elected vertex to \( VV_{123} \). The ambiguity is eliminated as shown in figures 15 to 18.

![Figure 15](image3.png)

**Figure 15:** Voronoi vertex vs. Voronoi edge case. Delaunay vertex \( DV_D \) elected as apex.

![Figure 16](image4.png)

**Figure 16:** Voronoi vertex vs. Voronoi edge case. Delaunay vertex \( DV_D \) elected as apex.

![Figure 17](image5.png)

**Figure 17:** Voronoi vertex vs. Voronoi edge case. Delaunay vertex \( DV_B \) elected as apex.

![Figure 18](image6.png)

**Figure 18:** Voronoi vertex vs. Voronoi edge case. Delaunay vertex \( DV_B \) elected as apex.
Algorithm 1 Solving Voronoi vertex vs. Voronoi edge Case

\[ [T_i, T_12] = \text{solveCaseVerVsEdge}(VV, V E) \]

**Input:**
- \( VV \): Voronoi vertex
- \( V E \): Voronoi edge

**Output:**
- \( T_i \): tetrahedron related to \( VV \)
- \( T_{12} \): set of at most two \( T_{12} \) tetrahedrons

**Precondition:**
- level of \( VV \) is not the same level of \( V E \);
- the projection of \( VV \) lies inside \( V E \)

**Postcondition:**
- the tetrahedrons in \( T_i \) and \( T_{12} \) form a valid configuration

1: \( Apex = \) elect left or right vertex of \( V E \)
2: \( T_i = \) new \( T_i \) using the Delaunay triangle related to \( VV \) and \( Apex \)
3: \( \text{for every Voronoi edge } V E_k \text{ related to } VV \text{ do} \)
4: \( \text{if half-plane of } V E_k \text{ is not the same half-plane of } Apex \text{ then} \)
5: \( t_{12} = \) new \( T_{12} \) created with the Delaunay edges related to \( V E_k \) and \( V E \)
6: \( \text{add } t_{12} \text{ to } T_{12} \)
7: \( \text{end if} \)
8: \( \text{end for} \)

**Identification of a valid configuration**

The complete sequence of steps is given in detail in algorithm 1. The infinite version of the Voronoi edge on which the vertex is projected, \( VE_{BD} \) in figure 17, divides the plane into two half-planes, each of them containing one of the Delaunay vertices related to the edge and one or two projected Delaunay edges belonging to level \( i \). Due to the virtual displacement done by electing the apex (line 1), the edges contained in the half-plane where the apex lies, do not longer intersect \( VE_{BD} \) but the edges in the second half-plane properly do it (line 4). This “intersect and no-longer-intersect” status on each found intersection leads to a proper creation of the nodes related to the \( T_{12} \) tetrahedrons that complete a valid configuration (lines 3-7).

**3.3.2 Case 2: Voronoi vertex vs. Voronoi Vertex**

This case is generated when six Delaunay vertices are co-spherical, leading to a Voronoi vertex belonging to level \( i \) be projected a Voronoi vertex belonging to level \( j \).

In this case nine intersections are identified, \( VE_{12} \) vs. \( V E_{AB} \), \( VE_{12} \) vs. \( V E_{BC} \), \( VE_{12} \) vs. \( V E_{CA} \), \( VE_{23} \) vs. \( V E_{AB} \), \( VE_{23} \) vs. \( V E_{BC} \), \( VE_{23} \) vs. \( V E_{CA} \), \( VE_{31} \) vs. \( V E_{AB} \), \( VE_{31} \) vs. \( V E_{BC} \), \( VE_{31} \) vs. \( V E_{CA} \). As in the previous case, the construction of these nine tetrahedrons leads to an inconsistent graph. The “election of apex” problem is also present, with the extrafact that there are two \( T_i \) tetrahedrons to elect an apex, and three possible apices for each tetrahedron. When an apex for any of the \( T_i \) tetrahedrons is chosen, it restricts the election of the apex for the second \( T_i \) tetrahedron and the creation of the complementary \( T_{12} \) tetrahedrons. Because
of this, three configurations are allowed as solutions in this case.

As in the previous case, the election of an apex could be translated into a virtual displacement of the levels and the elimination of the ambiguity by the assumption that the distance between the apex and the Voronoi vertex is the smallest.

Algorithm 2

**Identifying Vertex vs. Vertex sub-cases**

```plaintext
subcase = idVerVsVerSubcase(VV_i, VV_j)
```

**Input:**
- $VV_i$: Voronoi vertex on level $i$
- $VV_j$: Voronoi vertex on level $j$

**Output:**
- subcase: Flag indicating the sub-case type; its possible values are $1a2b3c$ or $1ab23c$

**Precondition:**
- Level of $VV_i$ is not the same level of $VV_j$;
- The projection of $VV_i$ lies on the projection of $VV_j$.

**Postcondition:**
- A sub-case is identified.

1.Edges[] = angular order of all edges related to $VV_i$ and $VV_j$
2.Subcase = $1a2b3c$
3.for every Voronoi edge $VE_c$ in Edges do
4.set $VE_e$ as the edge next to $VE_c$ in Edges
5. if level of $VE_c$ is level of $VE_e$ then
6. subcase = $1ab23c$
7. end if
8. end for

Two different sub-cases are identified for this case and both of then keep the same characteristics described above. Algorithm 2 identifies the sub-cases. The sub-cases are determined by the distribution of the edges on the “intersecting star” created when all the edges are projected on the same plane (see figure 19 to 22 ). There are only two possible distributions. (a) the edges are intercalated or (b) they are not. When two consecutive edges belong to the same level, the sub-case is identified as the $1ab23c$ sub-case (lines 5-7). If there are no two consecutive levels belonging to the same level, the sub-case is identified as the $1a2b3c$ sub-case (the cycle in lines 3-9 never falls inside lines 5-7).

**Identification of a valid configuration for the $1a2b3c$ sub-case**

For this sub-case, each Voronoi region related to a Voronoi vertex contains a Voronoi edge related to the other Voronoi vertex (figures 19 and 20 ). The three solutions for this sub-case are symmetric; the election of the apex for the first $T_i$ tetrahedron does not change the fact that two $T_{12}$ and two $T_i$ tetrahedrons are created. In figures 23 and 24 the Joint Voronoi Diagram with no ambiguity is shown, and also its physical tetrahedron representation.

Algorithm 3 implements the solution for this sub-case. The election of the apex for the first $T_i$ is performed in line 4. The vertex that lies in the region that is opposite to the first elected apex in the consecutive level is chosen as apex for the second $T_i$ tetrahedron (line 6-8). The $T_{12}$ tetrahedrons that complete the valid solution are created using the edges that bound the corresponding Voronoi regions of the elected apices (lines 11-13 and 15-17).

**Identification of a valid configuration for the $1ab23c$ sub-case**

In contrast with the previous sub-case, the solutions are not symmetric in this sub-case. The solutions are shown in figures 25 to 28.

The simplest solution is shown in figure 27, where just one $T_{12}$ tetrahedron is created. To construct that
Algorithm 3 Voronoi vertex vs. Voronoi vertex 1a2b3c sub-case

\[ [T_i, T_{12}] = \text{solveVerVsVer1a2b3c}(VV_i, VV_j) \]

**Input:**
- \( VV_i \): Voronoi vertex on level \( i \)
- \( VV_j \): Voronoi vertex on level \( j \)

**Output:**
- \( T_i \): set of TWO \( T_i \) tetrahedrons related to \( VV_i \) and \( VV_j \)
- \( T_{12} \): set of TWO \( T_{12} \) tetrahedrons

**Precondition:**
- level of \( VV_i \) is not the same level of \( VV_j \);
- the projection of \( VV_i \) lies on the projection of \( VV_j \)

**Postcondition:**
- the tetrahedrons in \( T_i \) and \( T_{12} \) form a valid configuration

1. \( Edge_j = \) any Voronoi edge related to \( VV_j \)
2. \( Region_j = \) Voronoi region to the left of \( Edge_j \)
3. \( Apex_1 = \) Delaunay vertex related to \( Region_j \)
4. \( t_i = \) new \( T_i \) using the Delaunay triangle related to \( VV_j \) and \( Apex_1 \)
5. add \( t_i \) to \( T_i \)
6. \( Edge_i = \) Voronoi edge whose projection lies inside \( Region_j \)
7. \( Region_i = \) Voronoi region not bounded by \( Edge_i \) on the level of \( Edge_i \)
8. \( Apex_2 = \) Delaunay vertex related to \( Region_i \)
9. \( t_i = \) new \( T_i \) using the Delaunay triangle related to \( VV_i \) and \( Apex_2 \)
10. add \( t_i \) to \( T_i \)
11. \( DE_i = \) Delaunay edge related to the Voronoi edge to the left of \( Region_i \)
12. \( DE_j = \) Delaunay edge related to the Voronoi edge to the right of \( Region_i \)
13. \( t_{12} = \) new \( T_{12} \) using \( DE_i \) and \( DE_j \)
14. add \( t_{12} \) to \( T_{12} \)
15. \( DE_i = \) Delaunay edge related to the Voronoi edge to the right of \( Region_i \)
16. \( DE_j = \) Delaunay edge related to the Voronoi edge to the left of \( Region_i \)
17. \( t_{12} = \) new \( T_{12} \) using \( DE_i \) and \( DE_j \)
18. add \( t_{12} \) to \( T_{12} \)

solution, some elements must be identified: the **Full Region** and the **Lone Edge**.

**Full Region:** The Voronoi region that contains two Voronoi edges belonging to the other level is named the **Full Region**. In figure 27, the **Full Region** for level \( i \) is the Voronoi region \( VR_1 \), bounded by \( VE_{12} \) and \( VE_{31} \), and for level \( j \) it is the Voronoi region \( VR_2 \), bounded by \( VE_{BC} \) and \( VE_{CA} \).

**Lone Edge:** The Voronoi edge that is alone in a Voronoi region belonging to the other level is called the **Lone Edge**. In figure 27, \( VE_{12} \) and \( VE_{BD} \) are the **Lone Edges** for levels \( i \) and \( j \) respectively.

Algorithm 4 formalizes the solution for this sub-case. The valid construction is composed by the \( T_{12} \) tetrahedron related to the intersection of both Lone Edges (line 15), and the \( T_i \) tetrahedrons created by each Delaunay triangle related to a Voronoi vertex and the Delaunay vertex related to the Full Region of the other level used as the apex (lines 7-11). In figure 28 the tetrahedrons related to this solution are shown.
Algorithm 4 Solving Vertex vs. Vertex 1ab23c sub-case

\[ [T_i, T_{12}] = \text{solveVerVsVer1ab23c}(VV_i, VV_j) \]

**Input:**
- \( VV_i \): Voronoi vertex on level \( i \)
- \( VV_j \): Voronoi vertex on level \( j \)

**Output:**
- \( T_i \): set of TWO \( T_i \) tetrahedrons related to \( VV_i \) and \( VV_j \);
- \( T_{12} \): a \( T_{12} \) tetrahedron

**Precondition:**
- level of \( VV_i \) is not the same as the level of \( VV_j \);
- the projection of \( VV_i \) lies on the projection of \( VV_j \)

**Postcondition:** Tetrahedrons in \( T_i \) and \( T_{12} \) form a valid configuration

1: \( \text{Level}_i = \text{level of } VV_i \)
2: \( \text{Edges}[\cdot] = \text{angular order of all edges related to } VV_i \text{ and } VV_j \)
3: \( [\text{Lone_edge}_i, \text{Lone_edge}_j] = \text{findLoneEdges}(\text{Edges, level}_i) \)
4: \( [\text{Full_region}_i, \text{Full_region}_j] = \text{findFullRegions}(\text{Edges, level}_i) \)
5: \( \text{base} = \text{Delaunay triangle related to } VV_i \)
6: \( \text{apex} = \text{Delaunay vertex related to } \text{Full_region}_j \)
7: \( t_i = \text{new } T_i \text{ using } \text{base} \text{ and } \text{apex} \)
8: add \( t_i \) to \( T_i \)
9: \( \text{base} = \text{Delaunay triangle related to } VV_j \)
10: \( \text{apex} = \text{Delaunay vertex related to } \text{Full_region}_i \)
11: \( t_j = \text{new } T_j \text{ using } \text{base} \text{ and } \text{apex} \)
12: add \( t_j \) to \( T_j \)
13: \( DE_i = \text{Delaunay edge related to } \text{Lone_edge}_i \)
14: \( DE_j = \text{Delaunay edge related to } \text{Lone_edge}_j \)
15: \( T_{12} = \text{new } T_{12} \text{ using } DE_i \text{ and } DE_j \)

4 ELIMINATION OF TETRAHEDRONS

The elimination of all the faces of the \( T_i \) tetrahedrons belonging to a non-solid connection leads to the creation of incomplete surfaces. This defect is fixed in the version implemented as part of this project. When a \( T_i \) tetrahedron belonging to a non-solid connection is eliminated, a hole results in the place where its base stood. To avoid such holes, the horizontal triangles (bases) of the \( T_i \) tetrahedrons eliminated by non-solid connections are kept and included into the reconstructed surface.

5 RESULTS

5.1 Skull

The Skull is a set of 258 contours, placed on 63 planes parallel to the XZ plane. The resulting surface is composed by 39,808 triangles. This set of contours presents wide \( m - n \) branches specially in the levels between the nose and the eye holes. In figure 29 a detail of levels \( i \) and \( i + 1 \) is shown. In figures 30 to 33 more details may be observed.

Figure 29: Detail of levels \( i \) and \( i + 1 \) of the set of contours “skull”

Figure 30: Set of contours
5.2 Brain

This set of contours is a complement given with the algorithm of the B+G method\textsuperscript{1}. It is composed by 15 parallel levels, with 105 contours. The reconstructed surface has 13,607 triangles. Figures 34 to 37 shows more details.

\textsuperscript{1}ftp://ftp-sop.inria.fr/prisme/NUAGES/Nuages/

6 CONCLUSIONS

A method has been designed and implemented, to circumvent geometrical degeneracies arising from si-
multaneous processing of 2D superimposed Voronoi Diagrams, in the context of Surface Reconstruction from Slice Samples. In this particular problem, for each degenerate condition an enumerable finite set of non-degenerate counterparts is programmed, and instantiated as the geometry of the degeneracy dictates. In absence of the algorithm, selfintersecting and therefore non-manifold constructions are produced. With the algorithm, degenerate cases are mapped to their non-degenerate counterparts. This allows the normal downstream execution of the host algorithm (B+G, by Boissonnat & Geiger, 1988, 1993). The method presented classifies actions to be taken, based on the level of the degeneracy. The results show that the method is successful in removing the degeneracy, without further iterations and in a deterministic way. This method can be applied when the number of cases of degeneracy is known.

REFERENCES