Validation and Storage of Polyhedra through Constrained Delaunay Tetrahedralization

Edward Verbree[1] and Hang Si[2]

[1] Delft University of Technology
Research Institute OTB, Section GIS-technology
Jaffalaan 9, 2628BX Delft, the Netherlands
e.verbree@tudelft.nl
[2] Weierstrass Institute for Applied Analysis and Stochastics
Research Group: Numerical Mathematics and Scientific Computing
Mohrenstr. 39, 10117 Berlin, Germany
si@wias-berlin.de

Abstract. Closed, watertight, 3D geometries are represented by polyhedra. Current data models define these polyhedra basically as a set of polygons, leaving the test on intersecting polygons or open gaps to external validation rules. If this testing is not performed well, or not at all, non-valid polyhedra could be stored in geo-databases. This paper proposes the utilization of the Constrained Delaunay Tetrahedralization (CDT) for the validation (i.e. check on self-intersecting and closeness) of polyhedra on the one hand, and the efficient storage of valid polyhedra on the other hand. The paper stresses on the decomposition of a polyhedron through a CDT and the possibility to store and compose the polyhedron through the vertices of the CDT, a bitmap that indicates which faces of the Delaunay Tetrahedralization (DT) links to a CDT-face, and a list of non-recovered CDT-faces.

1 Introduction

Real world objects are characterized by a particular representation, identified and captured, and stored according a specified datamodel in a geo-database. For applications within the 3D domain, one cannot work any longer with 'down to earth' flattened objects, which are defined as polygon footprints attached to a 2D or 2.5D surface. An appropriate 3D representation is needed, which has to be supported by a suitable 3D data model.

The polyhedral approach, as described in [1] defines the boundary of a 3D primitive (or simple object) as a set of polygons, where the vertices of each polygon are coplanar and valid according the Simple Feature Specification of the OGC [2], i.e. non self-intersecting and closed. The polygons, defining the boundary of the 3D primitive, should be connected to each other in such a way that the 3D primitive itself is 'closed' (or 'watertight'). That yields the set of polygons consist of non mutual-intersecting polygons and does not leaves open gaps, resulting in a connected interior for the 3D primitive (simple object).

T.J. Cova et al. (Eds.): GIScience 2008, LNCS 5266, pp. 354–369, 2008.

1.1 Data Models for Polyhedra

Three-dimensional geometries (simple solids) have to be stored in a geo-database according to a supporting data model. In [3] a review is given of the ISO 19107 Schema [4], OGC GML specification [5] and Oracle Spatial SDO_GEOMETRY data type to store 3D geometries. This paper [3] defines more specific and refined rules for valid geometries. In relation to standardization organizations it states: "ISO and OGC have tried to give unambiguous and complete definitions of valid geometric primitives. But as it was already pointed out in [6] it turns out that the standards are not unambiguous and complete, even in the case of 2D-polygons. For 3D geometric primitives there is an abstract ISO specification [4], but this is not the needed implementation specification".

The GML and ISO definition for solids reads: "A solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces (shells). A shell is represented by a composite surface, where every shell is used to represent a single connected component of the boundary of a solid. It consists of a composite surface (a list of orientable surfaces) connected in a topological cycle. Unlike a ring, a shell's elements have no natural sort order. Like rings, shells are simple. The element 'exterior' specifies the outer boundary of the solid. Boundaries of solids are similar to surface boundaries. In normal 3-dimensional Euclidean space, one (composite) surface is distinguished as the exterior. The element 'interior' specifies the inner boundary of the solid."

In [3] a description of Simple Solids (Polyhedra) in Oracle is given: "A simple solid is defined as a 'Single Volume' bounded on the exterior by one exterior composite surface and on the interior by zero or more interior composite surfaces. To demarcate the interior of the solid from the exterior, the polygons of the boundary are oriented such that their normal vector always point "outwards" from the solid."

The Computational Geometry Algorithms Library (CGAL) [7] gives the following work-definition of a polyhedron: "Polyhedral surfaces in three dimensions are composed of vertices, edges, facets and an incidence relationship on them. The organization beneath is a halfedge data structure, which restricts the class of representable surfaces to orientable 2-manifolds - with and without boundary. If the surface is closed we call it a polyhedron. Each edge is represented by two halfedges with opposite orientations. Facets are defined by the circular sequence of halfedges along their boundary. The halfedges along the boundary of a hole are called border halfedges and have no incident facet. An edge is a border edge if one of its halfedges is a border halfedge. A surface is closed if it contains no border halfedges. A closed surface is a boundary representation for polyhedra in three dimensions."

1.2 Validation of Polyhedra

Surprisingly, existing schemas, specifications and even implementations define data-models to store 3D geometries, but they are not limited to store valid, thus 'watertight', 3D geometries only: also 'non-watertight' (boundary intersecting, open, non-connected interior) 3D geometries can be stored. In [3] the following validation rules/tests of solids are defined. These tests check on both a valid, closed boundary as on the connectedness of the interior.

These operations should use tolerance values to deal with the problems to express exact fractions in decimal notion with a finite number of binary digits.

- Single Volume check: the volume should be contiguous:
  - Closed test: the boundary has to be closed.
  - Connect test: the volume has to be connected. This means each component of the solid should be reachable from any other component.
- Inner-outer check:
  - Every surface marked as an inner boundary should be 'inside' the solid defined by the exterior boundary.
  - Inner boundaries may never intersect but only touch under the condition that the interior of the solid remains connected.
- Orientation check: The volume bounded by the exterior boundary is computed as a positive value if every face is oriented such that each normal is pointed away from the solid due to the Greens Theorem. Similarly, the volume bounded by the interior boundary is computed as a negative value. If each exterior and interior boundary obeys this rule and they pass the connect test as well, then this check is passed.
- Element-check: Every specified surface is a valid surface.

A possible way to avoid non-valid polyhedra, i.e. self-intersecting faces (a non circular sequence of halfedges) or non-planar faces, is to restrict the boundary of the polyhedra to be composed of a set of triangular facets. But even then this 'solid object' has to be validated, as it is still possible that the triangular facets of this 'polyhedron' could intersect each other.

1.3 Storage of Polyhedra through Tetrahedralization

The idea to use a tetrahedralization to represent polyhedra is based on previous work by the first author in representing Planar Maps through Conforming Delaunay Triangulations [8]. A method was described to store (encode) and receive (decode) a Planar Map (PM) through a Constrained Delaunay Triangulation (CDT) with applications in a server-client environment. Planar maps are embeddings of topological maps into the plane. A planar map subdivides the plane into vertices, edges, and faces [9]. In two dimensions a true, conforming, Delaunay Triangulation that constrains to the input can be created by adding Steiner points at the edges of the Planar Map.

To store a PM the server creates a CDT of the edges of the PM. As the PM is now embedded by the CDT it is sufficient to send to the client the list of coordinates of the CDT nodes and an efficient encoded bitmap of the corresponding PM-CDT edges. The client determines a Delaunay Triangulation (DT) of the received list of coordinates of the CDT nodes. The DT at the client side is - omission degenerated cases - equivalent to the CDT at the server side. The edges of the PM are recovered within this DT by the bitmap (true/false) of the corresponding PM-CDT edges.

Within [8] it was stated: "Despite all these considerations, the encoding (decomposition) of Planar Maps by Conforming Delaunay Triangulations could be extended to the third dimension. Polyhedron boundary representations could be encoded (decomposed) and decoded (composed) through a conformal tetrahedralization". That idea

was presented in [10]. But, although polyhedra can be represented by a Conforming Delaunay Tetrahedralization, this can result in an extraordinary amount of added Steiner Points. In two-dimensions [11] shows an example that two-dimensional conforming Delaunay triangulations may need a quadratic number of Steiner points with respect to the input number of nodes. Therefore, we have adapted the method in this paper by applying Constrained Delaunay Tetrahedralization.

This paper links up with research by Penninga at al. on a simplicial complex-based DBMS approach to 3D topographic data modeling' [12] that has a focus on updating features in a TEN-based DBMS approach [13].

1.4 Our Work

We propose the use of a Constrained Delaunay Tetrahedralization (CDT) to validate and store polyhedra. By definition a polyhedron has to have one closed, watertight, polygonal outer surface and possible one or more inner surfaces. The validation of this important property has to be done in advance, as most geo-databases allow the storage of polygonal surfaces and polyhedra through the same datamodel. To check whether or not a polygonal surface is closed, and thus watertight, is not a straightforward task, as even the existing standards and specifications defining polyhedra (simple solids) are ambiguous.

The motivation behind the use of a CDT is given by its construction scheme. First and for all, if a polyhedron is valid, the faces of the derived CDT have to match the faces of its surface triangulation completely. This requirement holds also for polygonal surfaces, but as the final phase of the tetrahedralization removes the tetrahedra outside the surface triangulation, only closed, watertight, polyhedra survive.

The second advantage of applying a CDT is given by the possibility to store the geometry of the polyhedron by only its nodes, the added Steiner points, a Delaunay Tetrahedralization (DT) of these nodes and Steiner points, a bitmap of the lexical ordered faces (natural sorted on three nodes) of this DT (thus only one bit per face), and a list of the DT missing triangles of its surface triangulation. The efficiency of this approach depends on one hand on the amount of added Steiner points and on the other hand on how much the CDT conforms to the DT.

1.5 Outline

In the preliminary section of this paper (section 2) we describe the possibilities and limitations of current datamodels for the storage and validation of polyhedra. We give formal definitions of Conforming Delaunay Tetrahedralization and Constrained Delaunay Tetrahedralization. In section 3 we focus on the issue of validating polyhedra by means of CDT and we discuss the possibility to detect self-intersecting polygons and the distinction between (closed, watertight) polyhedra and polygonal surfaces. Section 4 describes the possibility to store polyhedra through CDT and some examples and test results on the efficiency of this approach are given. Section 5 resumes with the conclusions.

2 Preliminaries

This section presents the essential definitions and notations of the geometric objects used in this paper. These objects are based on the fundamental concepts developed in topology and geometry. Good introductory texts are given by Edelsbrunner [14] and Ziegler [15].

2.1 Polyhedra and Faces

In this section, we define a general and therefore not necessarily convex polyhedron and its faces.

Definition 1 (Polyhedron). A *polyhedron* P in R^d is the union of a finite set P of convex polyhedra, i.e., $P=\cup_{U\in P}U$, and the space of P is connected.

The *dimension* dim(P) is the largest dimension of a convex polyhedron in P. Note that P may contains holes in its interior. Whatever, we require that the space of P must be connected, i.e. any two points in the interior of P can be connected through a path in the interior of P. See Fig. 1 for examples.

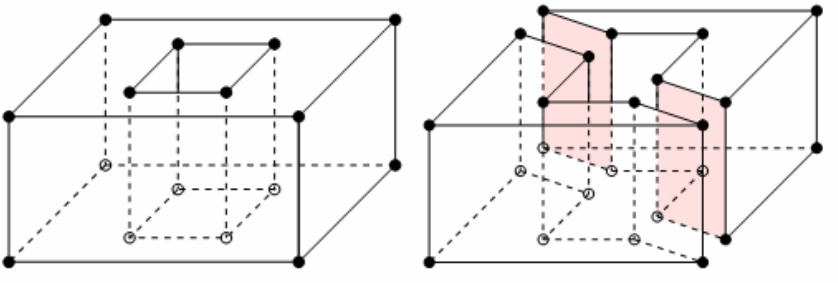


Fig. 1. Polyhedra and faces. Left: A three-dimensional polyhedron (a torus) formed by the union of four convex polytopes. It consists of 16 vertices (zero-faces), 24 edges (1-faces), 10 two-faces (the faces at top and bottom are not simply connected), and 1 three-face (which is itself). Right: Two three-dimensional polyhedra. Each one has 12 vertices, 18 edges, 8 two-faces, and 1 three-faces. The shaded area highlights two 2-faces whose points have the same face figures.

There are few definitions about faces of a non-convex polyhedron. The following definitions of faces are mainly by [14] with only difference in the connectness of the faces. Let $B(\mathbf{x},r)$ denote the open ball in R^d centered at $\mathbf{x}$ with radius r.

For a point $\mathbf{x}$ in a polyhedron P we consider a sufficiently small neighborhood $N_\varepsilon(\mathbf{x})=(\mathbf{x}+B(\mathbf{0},\varepsilon))\cap P$. The *face figure* of $\mathbf{x}$ is the enlarged version of this neighborhood within P, i.e., $\mathbf{x}+\cup_{\lambda>0}\lambda(N_\varepsilon(\mathbf{x})-\mathbf{x})$.

Definition 2 (Face). A *face* F of a polyhedron P is the closure of a maximal connected set of points with identical face figures.

By this definition, a face of P may contain holes in its interior, but it is always connected. See Fig. 1 for examples.

A face F of P is again a polyhedron. Particularly, $\varnothing$ is a face of P. If all convex polyhedra in P have the same dimension, then P itself is a face of P. All other faces of P are *proper* faces of P. We also write $F \leq P$ or $F < P$ if F is a face or a proper face of P. The faces of dimension 0, 1, $dim(P)-2$, and $dim(P)-1$ are called *vertices*, *edges*, *ridges*, and *facets*, respectively. The set of all vertices of P, the *vertex set*, will be denoted by $vert(P)$. The set of all proper faces of P is called the *boundary complex* of P, denoted as $bd(P)$. The *interior* $int(P)$ is $P - bd(P)$.

Note that a polyhedron may be unbounded. In the scope of this work, we always assume that a polyhedron is bounded, i.e., it is a polytope. In our later discussions, a polytope can be convex or non-convex.

2.2 Delaunay Triangulation / Tetrahedralization

Let S be a finite set of points in R^d. A *Delaunay triangulation* [16] (abbreviated as DT) of S is a triangulation T such that *(i)* the vertex set of T is S, *(ii)* the convex hull of S equals to the underlying space of T, and *(iii)* (Delaunay criterion) every simplex (i.e. triangle in 2D-space, tetrahedron in 3D-space) of T has a circumscribed ball whose interior contains no points of S. **Fig. 2** illustrates a two-dimensional Delaunay triangulation. The DT of a three-dimensional point set is also called a *Delaunay tetrahedralization*.

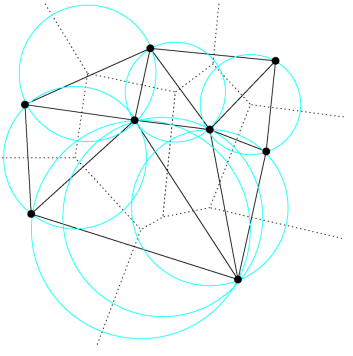


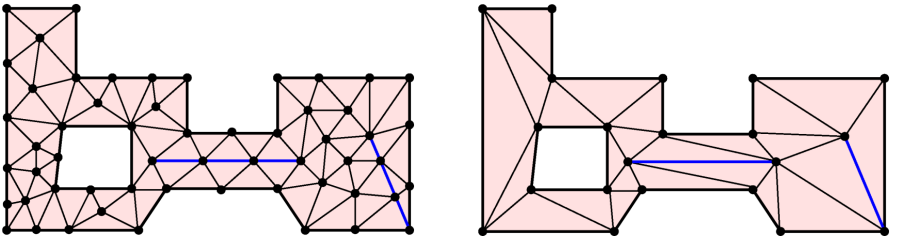
Fig. 2. The Delaunay triangulation (shown in solid black lines) of a two-dimensional point set. All circumcirles (shown in blue) of the triangles are empty. The dotted lines show the dual Voronoi diagram.

2.3 Conforming Delaunay Triangulation

Let P be a polygon. The DT of the vertices of P does not necessarily contain all the boundaries of P. A *conforming Delaunay triangulation* T of P is a Delaunay triangulation, such that

(i) each vertex of P is a vertex of T,
(ii) every boundary of P is a union of simplices of T, and
(iii) every simplex of T satisfies the Delaunay criterion.

Usually, a conforming DT of P will contain some additional points (so-called *Steiner points*) which do not belong to P (See Fig. 3 left for an example). It is worth to mention, even for a simple polygon, a large number of Steiner points may be needed. Let n be the number of points in the polygon. A simple example which needs $\Omega(n^2)$ Steiner points is given in [11].



Conforming DT Constrained DT

Fig. 3. Left: A conforming Delaunay triangulation of a two-dimensional Polygon P. Right: A constrained Delaunay triangulation of P.

2.4 Constrained Delaunay Triangulation

An alternative approach to form a triangulation of a polygon P is to use a Delaunay-like triangulation (not necessary to be a DT) which respects the boundaries of P.

The visibility between two vertices $\mathbf{p},\mathbf{q} \in P$ is defined as follows: **p** and **q** are *invisible* to each other if the line segment **pq** intersects with any boundary of P at an interior point of **pq**.

Let T be a triangulation such that the underlying space of T is the convex hull of the vertices of P. A simplex $\sigma \in T$ is *constrained Delaunay* if there exists a circum circle B_σ of σ such that either B_σ contains no other vertices of T or there is no vertex inside B_σ which is visible from the interior of σ. Then T is a *constrained Delaunay triangulation* (abbreviated as CDT) of P if

(i) Every boundary of P is represented by a union of simplices of T.
(ii) Every simplex of T is constrained Delaunay.

A two-dimensional CDT is illustrated in Fig. 3 at the right. A three-dimensional CDT is also called a *constrained Delaunay tetrahedralization*.

The above definition implies that a CDT of P may contain Steiner points, i.e., points which do not belong to P. When it is the case, we call it a *Steiner* CDT. Otherwise, it is called a *pure* CDT. It is well known that a pure CDT of a polyhedron may not exist. For instance, given the Schönhardt polyhedron [18], the problem to decide whether a pure CDT of P exists or not, is NP-complete [19]. On the other hand, there are infinitely many Steiner CDTs of P.

As in the 2-Dimensional case, many Steiner points can be needed to obtain a *genuine* Conforming Delaunay Tetrahedralization, see i.e. [20], [21], [22].

If Steiner points are allowed, Chazelle [23] showed that any simple polyhedron of n vertices may need $O(n^2)$ Steiner points, and this bound is tight in the worst case. Chazelle and Palios [24] presented an algorithm to decompose a simple polyhedron using $O(n+r^2)$ Steiner points, where r is the number of reflex edges (a quantitative measure of nonconvexity) of P . However, even for a simply shaped polyhedron, i.e. Fig. 4 (top-left) this algorithm will introduce unnecessarily large number of Steiner points, see Fig. 4 (top-right). More practical approaches using conforming Delaunay triangulations [20], [21] and constrained Delaunay triangulations [25], [26] are proposed, see Fig. 4 (bottom-left) and (bottom-right). However, no polynomial upper bound on the number of Steiner points is known; see the 22nd open problem in [14].

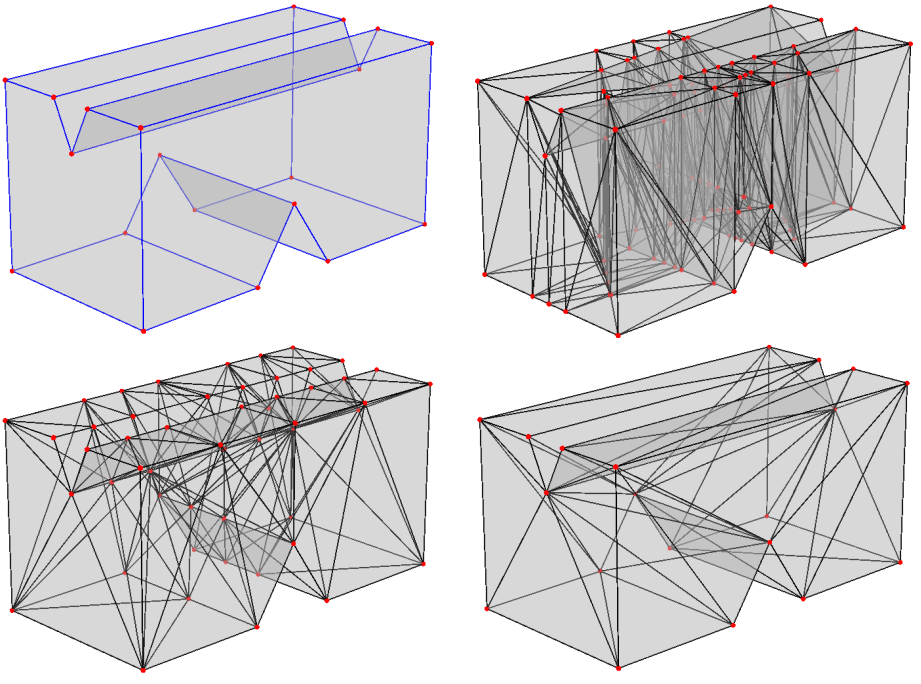


Fig. 4. Comparison of various approaches in decomposition (tetrahedralization) of a polyhedron: A 'simple' polyhedron with 20 vertices and 2 reflex can end up with 51 vertices and 103 tetrahedra to create a Conforming Delaunay Tetrahedralization

3 Validation of Polyhedra through CDTs

The data models presented in Section 1.1 do have one feature in common: the parts (i.e. polygons, shells, facets) of a three-dimensional solid are given. However there is no guarantee these parts are bound together to form a valid polyhedron. Various approaches have been proposed for validating a polyhedron from a given representation, see e.g., [3]. In this section, we propose a new approach for validating a polyhedron.

3.1 Decomposition of Polyhedra through Tetrahedralization

Our approach is based on this simple observation: A valid polyhedron P can be decomposed into a tetrahedralization T such that the boundary of P is represented as a union of simplices of T. This observation can be easily proved: Since P is valid, then let T be a Constrained Delaunay Tetradedralization (CDT) of P. Otherwise, if such a T does not exist, then either the boundary *bd(P)* of P contains self-intersections or *bd(P)* is not closed. Hence, the problem of validating a polyhedron P can be transformed into the problem of finding a tetrahedralization of P. This is a long studied problem in computational geometry (see e.g., [23], [24], [27]), as well as in mesh generation (see e.g. [28], [29], [30]).

To validate P takes two steps: *(1)* check the self-intersection in the boundary of P, and *(2)* generate the CDT of P. Note that step *(2)* can be called only if step *(1)* is successful. Then P is valid only if a CDT of P can be generated, the boundary of P is represented as a union of simplices of T, and the interior of T is connected.

The generation of CDTs is an active research topic. The discussion of such methods is out of the scope of this work, we refer to the research paper of Shewchuk [25] and Si et al [26]. A software implementation of the CDT algorithms can be found in the program `TetGen` [31]. In the following, we briefly discuss the approach to detect self-intersections.

3.2 Self-intersection Detect

A basic requirement for a 3D polyhedron P is that any two facets of P should touch only along their common faces. Let T_1 and T_2 be two triangles in R^3, let $U=T_1 \cap T_2$. We say that T_1 and T_2 *intersect* each other if $U \neq \varnothing$ and U is not a proper face of both T_1 and T_2. See Fig. 5 for examples.

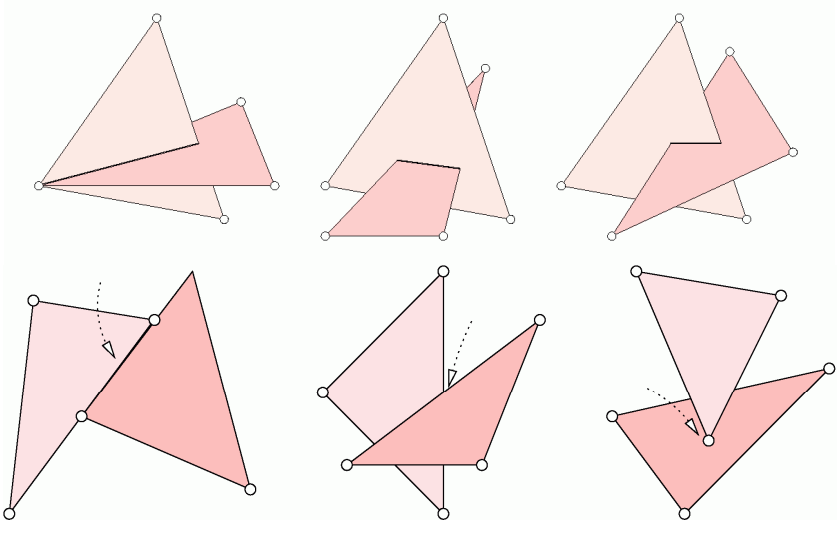


Fig. 5. Possible cases of two triangles in R^3 can intersect each other.

The problem. Consider the set of triangles forming the triangulation F of the boundary $bd(P)$ of P, find all intersected pairs of triangles in F.

The primitive operation for this task is a function *TRI_TRI_INTERSECT*(T_1,$T2$). That is, it takes two triangles T_1,T_2 in R^3 as inputs, return a value indicating whether T_1 and T_2 intersect each other or not.

Fast algorithms are proposed to test the intersection between three-dimensional triangles, such as Möller [32] and Guigue et al [33]. In practice, Möller's algorithm is less robust since it needs to compute an intermediate line interval which requires arithmetic precision. Guigue et al's algorithm relies exclusively on orientation predicates, which eliminates the intermediate error caused by floating-point arithmetics. However, both algorithms do not satisfy our goal. Let $U=T_1 \cap T_2$. The problem lies that both algorithm only detect whether U is empty or not. In our problem, we need to distinguish more cases when $U \neq \varnothing$. To distinguish these cases needs to handle all degenerate cases.

We implemented a triangle-triangle test algorithm. The idea is similar to that of Guigue et al [33], only the three-dimensional orientation test is involved. It further classifies all degenerate cases based on the study of the signs. A trivial approach to check the boundary self intersection of the polyhedron is just to test the intersection of triangles pair by pair, which takes $O(m^2)$ time, where m is the number of triangles. A simple way to improve the speed is to filter out triangle pairs that can not intersect by first doing the intersection of their bounding boxes whose sides are parallel to the axes. It has been shown [34] that such box intersection can be reported in $O(m\log^2 m+J)$ time, where J is the number of intersected pairs.

We implemented a divide-and-conquer approach for reducing the number of intersection tests. Starting from the bounding box of the vertices, it recursively partitions the box into smaller boxes, until the number of triangles in a box is not decreased anymore. Then a brute-force pairwise triangle-triangle intersection test is performed. This approach runs in time $O(m\log m+I^2)$, where I is the largest number of triangles appearing in a box which is not partitioned. It is possible that $I=m$, i.e., the worst-case time complexity is $O(m^2)$. For most inputs which have a dense point set, I is relatively small comparing to m. Fig. 6 illustrates an example.

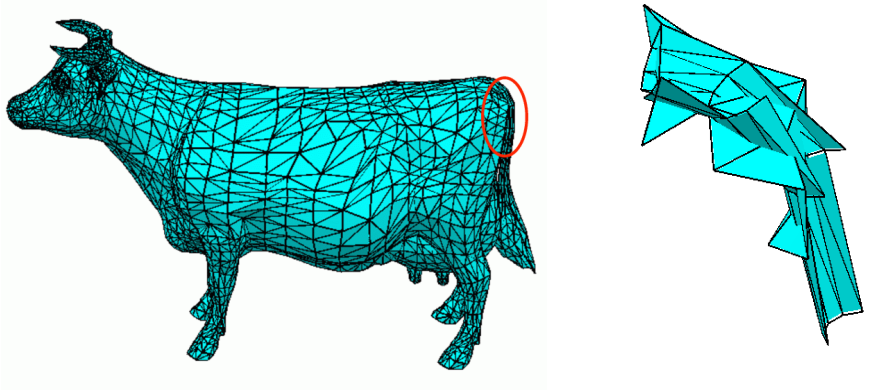


Fig. 6. Boundary intersection check. A surface mesh of a Cow is shown left. The triangles on the tail (the annotated place) are intersecting each other shown on the right.

4 Storage of Polyhedra through Constrained Delaunay Tetrahedralization

To proof the efficiency (i.e. gain of storage) of decomposition (encoding) and composition (decoding) a polyhedron through means of a tetrahedralization is not trivial.

The basic idea of this method is the decomposition of a given polyhedron by a Constrained Delaunay Tetrahedralization (CDT) of the nodes of the polyhedron. As shown in section 2, extra Steiner points are needed to make sure each polygon of the polyhedron is represented by a set of faces of the CDT. Within the obtained CDT, the faces which 'belong' to the polyhedron are marked with a binary bit '1', and the faces of the CDT that do not 'belong' to the polyhedron (thus faces inside or outside the polyhedron) are marked with a binary bit '0'. The composition is performed by a regular Delaunay Tetrahedralization (DT) of the original nodes of the given polyhedron plus the extra Steiner points.

4.1 Implementation Issues

One complication within this process is that the set of faces of the DT created during the composition phase can be different of the set of faces of the CDT created during the decomposition phase. The first reason for this is because the Constrained DT (CDT) does not have to be a Conforming Constrained Delaunay Tetrahedralization (CCDT): a CDT-tetrahedron does not have to obey in general to the Delaunay-criterion of an empty circumsphere. One option to work around that problem is to create during the decomposition phase a CCDT also, as suggested in [10]. But in creating a CCDT one needs to add far more Steiner points [20], [21], [22] and this extra amount of Steiner points limits this method. The second reason is caused by possible degenerated cases: five or more points directly on one circumsphere can be tetrahedronized in a different way by another DT-algorithm. So, even if a CCDT can be created, the DT of the same set of input points can result in another set of tetrahedra. One possibility to 'solve' this problem is to define a unique decision rule, i.e. two preferred directions as presented in for 2-dimensional DT in [35]. This method could be extended to a Delaunay Tetrahedralization, but this will cause some issues in defining directions in 3D.

For this reason, the coding of the faces 'belonging' to the polyhedron is based on the DT. One binary bit is needed for each face of the DT. A DT-face is to be marked (flagged) with a binary bit '1' when the corresponding CDT-face is marked (flagged) with a '1' and thus the DT-face is part of the boundary of the polyhedron. All other DT-faces are to be marked with a '0' to identify these faces as part of the interior or to be outside the polyhedron. The 'bitmap' of marked and non-marked lexical ordered faces (i.e. natural sorted on their nodes) is stored. To emphasize, the DT faces themselves are not stored, but only a bit for each face indicating whether or not the face is part of the boundary of the polyhedron. The CDT-faces marked with a '1', but not part of the DT, have to be stored separately.

In summery, the decomposition process consists of the following steps:

1. Create a CDT of the given polyhedron *P;*
2. Store CDT-nodes (hence nodes of polyhedron and extra Steiner points);

3. Mark CDT-faces (‘1’: face is part of boundary of polyhedron *P*, ‘0’ remaining faces);
4. Create DT of nodes CDT;
5. Mark DT-faces (‘1’: DT-face that corresponds to CDT ‘1 ‘ face is identical marked ‘1’; ‘0’ remaining faces);
6. Store ‘bitmap’ of marked and non-marked ordered DT-faces;
7. Store ‘1’-marked CDT-faces which do not have a corresponding DT-face.

The composition process consists of the following steps:

1. Read CDT-nodes;
2. Create DT of CDT-nodes;
3. Read ‘bitmap’
4. Order DT faces and set each DT-face according to the bitmap;
5. Create polygonal surface out of ‘1’-marked DT-faces;
6. Read non-recovered CDT-faces;
7. Create a watertight polyhedron boundary by completing the polygonal surface with non-recovered CDT-faces.

The gain in storage is within the difference of storing the polyhedron just by its nodes and the faces of its surface at the one hand, and the storage of the nodes of the CDT and the non-recovered CDT-faces at the other hand. To make this comparison more easy we assume the polyhedron is defined by a triangulated surface, thus all faces of the polyhedron are triangles. The storage requirements for this kind of triangular polyhedra are as follows: for each node of the polygon we need 3 floats for the X, Y and Z-coordinate, and for each face 3 integers (reference value to nodes).

The triangulated boundary of the polyhedron makes the comparison more easy, but is introduces one main disadvantage as is also stressed in section 1.2 of [25]: each triangular face of the boundary is now constrained, and thus more Steiner points are needed to make sure the CDT faces are part of the boundary of the polyhedron.

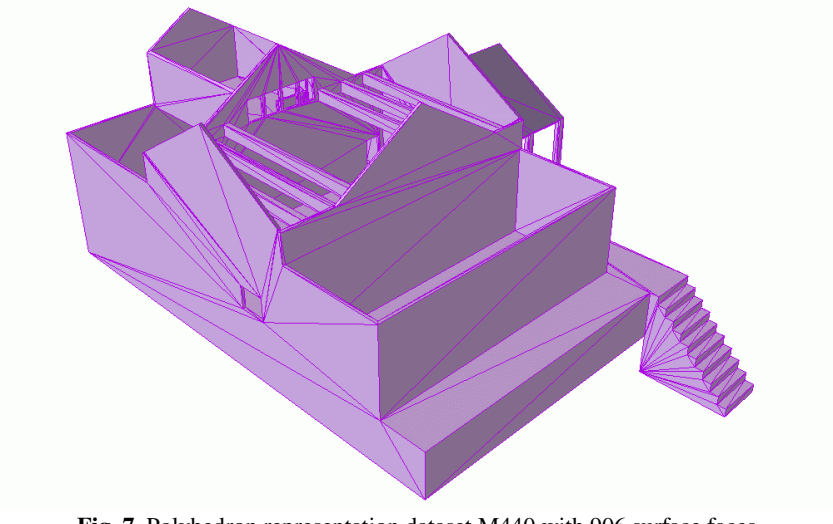


Fig. 7. Polyhedron representation dataset M440 with 906 surface faces

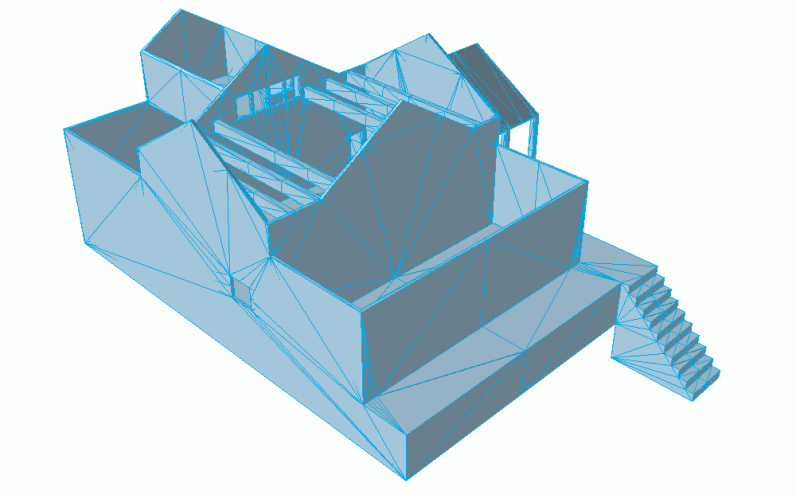


Fig. 8. Constrained Delaunay Tetrahedralization of Polyhedron representation of dataset M440

The storage requirements for the CDT encoded polyhedron sums to: for each node and also for all necessary Steiner points we need 3 floats for the X, Y and Z-coordinate, for each DT-face 1 bit, and for each non-recovered CDT-face 3 integers (reference values to nodes). As both methods require the storage of the nodes of the polygon, these are to be left out the comparison.

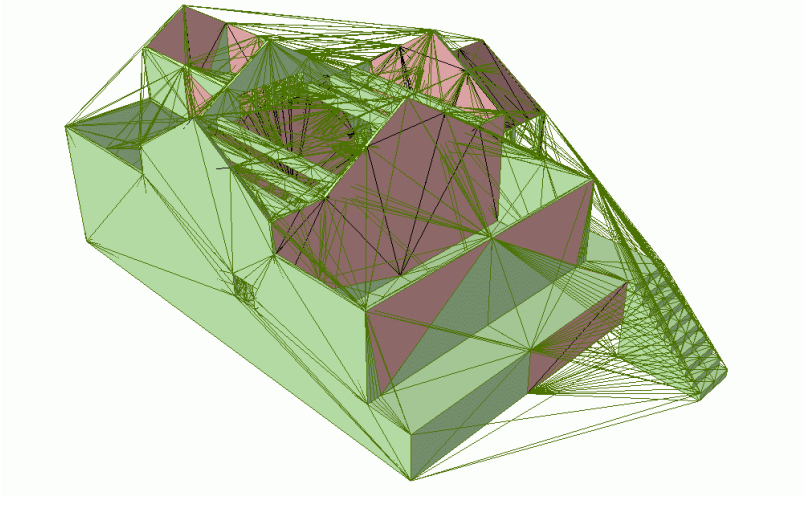


Fig. 9. Delaunay Tetrahedralization (DT) of vertices of Constrained Delaunay Tetrahedralization (CDT) of Polyhedron representation of dataset M440. The DT consists of 10545 faces. 187 Faces of the CDT are not recovered by the DT.

4.2 Experimental Results

We have tested this decomposition/composition method with some datasets from INRIAs 3D Meshed Research Database (`http://www-c.inria.fr/gamma/`). The tetrahedralization is performed by Tetgen 1.4.2 [31].

Polyhedron M440 represents a house, see Fig. 7. To store the 906 surface faces of the input model we need 906 * 3 * 32 = 86976 bits. The decomposition through means of a CDT as shown in Fig. 8, adds 480 Steiner Points, thus 480 * 3 * 32 = 46080 bits. The DT from the nodes of this CDT consists of 10545 faces (bits). The DT could not recover 187 faces of the CDT (see Fig. 9) so an extra 187 * 3 * 32 = 17952 bits are needed. In total we need: 46080 + 10545 + 17952 = 74577 bits. In conclusion, here we 'win' 86976 - 74577 = 12399 bits, or 14%.

For this dataset the gain is very modest. As mentioned in section 4.1, this result could be influenced by the triangulated boundary of the input polyhedron, as each face is to be recovered in the CDT.

5 Conclusions

We have demonstrated the utilization of the Constrained Delaunay Tetrahedralization (CDT) for the validation and efficient storage of polyhedra. The main advantage of this idea is not on the storage gain, but on the confidence to store closed, watertight polyhedral surfaces, and thus polyhedra. If a polyhedron is decomposable by a CDT, the surface faces of this CDT are to be connected to the surface of the polyhedron, and the interior of the CDT is connected, then the polyhedron is valid.

Further research could address the following questions:

- To what extent has a CDT to conform to the generic Delaunay Tetrahedralization? Adding more Steiner points can result in a more conforming or even completely Conforming Constrained Delaunay Tetrahedralization (CCDT). The efficiency is a trade off between this CCDT-likeness (with less non-CDT recovered faces to store) and the needed space to store the extra Steiner points and amount of faces of the Delaunay Tetrahedralization of the vertices of the CDT.
- What are the complications on validating and storing large datasets with many polyhedra collectively through one tetrahedralization? Is it still possible to validate a non-connected polyhedron or polyhedron completely inside another polyhedron? What will be the storage gain by the method as presented in this paper?

Acknowledgements

This research is carried out as part of the Dutch BSIK RGI program on 3D Topography (RGI-011). The authors like to express their gratefulness towards the reviewers of the GIScience 2008 conference and above all to Peter van Oosterom, Hugo Ledoux, Friso Penninga and Theo Tijssen for their rewarding comments.

References

[1] Arens, C., Stoter, J., van Oosterom, P.: Modelling 3D spatial objects in a geo-dbms using a 3D primitive. Computers & Geosciences 31, 165–177 (2005)

[2] OGC: Implementation specification for geographic information - simple feature access (2006), http://www.opengeospatial.org/standards/sfa

[3] Kazar, B.M., Kothuri, R., van Oosterom, P., Ravada, S.: On valid and invalid three-dimensional geometries. In: Advances in 3D Geoinformation Systems (2008)

[4] ISO: 19107 (2003), http://www.iso.org

[5] GML: The geographic markup language specification, version 3.3.1 (2003), http://www.opengeospatial.org/standards/gml

[6] van Oosterom, P., Quak, W., Tijssen, T.: About invalid, valid and clean polygons. In: Fisher, P.F. (ed.) Developments in Spatial Data Handling, 11th International Symposium on Spatial Data Handling, pp. 19–48 (2004)

[7] Kettner, L.: CGAL Manual. In: 3D Polyhedral Surfaces, ch. 12 (2008), http://www.cgal.org/Manual/3.3/doc_html/cgal_manual/Polyhedron/Chapter_main.html

[8] Verbree, E.: Encoding and decoding of planar maps through Conforming Delaunay Triangulations. In: ISPRS Workshop on multiple representation and interoperability of spatial data (2006)

[9] CGAL: Computational Geometry Algorithms Library (2007), http://www.cgal.org

[10] Verbree, E.: Piecewise linear complex representation through Conforming Delaunay Tetrahedronization. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 385–388. Springer, Heidelberg (2006)

[11] Edelsbrunner, H., Tan, T.S.: An upper bound for conforming Delaunay triangulations. SIAM Journal on Computing 22, 527–551 (1993)

[12] Penninga, F., van Oosterom, P.: A Simplicial Complex-based DBMS Approach To 3D Topographic Data Modelling. International Journal of Geographical Information Science 22, 751–779 (2008)

[13] Penninga, F., van Oosterom, P.: Updating Features in a TEN-based DBMS approach for 3D Topographic Data Modelling. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 147–152. Springer, Heidelberg (2006)

[14] Edelsbrunner, H.: Geometry and topology for mesh generation. Cambridge University Press, Cambridge (2001)

[15] Ziegler, G.M.: Lectures on Polytopes. Graduate Texts in Mathematics, vol. 152. Springer, New York (1995)

[16] Delaunay, B.N.: Sur la sphère vide. Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk 7, 793–800 (1934)

[17] Aurenhammer, F.: Voronoi diagrams – a study of fundamental geometric data structure. ACM Comput. Surveys 23, 345–405 (1991)

[18] Schönhardt, E.: Über die zerlegung von dreieckspolyedern in tetraeder. Mathematische Annalen 98, 309–312 (1928)

[19] Ruppert, J., Seidel, R.: On the difficulty of triangulating three-dimensional non-convex polyhedra. Discrete and Computational Geometry 7, 227–253 (1992)

[20] Murphy, M., Mount, D.M., Gable, C.W.: A point-placement strategy for conforming Delaunay tetrahedralizations. In: Proc. 11th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 67–74 (2000)

[21] Cohen-Steiner, D., De Verdière, E.C., Yvinec, M.: Conforming Delaunay triangulation in 3D. In: Proc. 18th annual ACM Symposium on Computational Geometry (2002)
[22] Cheng, S.W., Poon, S.H.: Graded conforming Delaunay tetrahedralization with bounded radius-edge ratio. In: Proc. 14th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 295–304 (2003)
[23] Chazelle, B.: Convex partition of a polyhedra: a lower bound and worest-case optimal algorithm. SIAM Journal on Computing 13, 488–507 (1984)
[24] Chazelle, B., Palios, L.: Triangulating a nonconvex polytope. Discrete and Computational Geometry 5, 505–526 (1990)
[25] Shewchuk, J.R.: General-dimensional constrained Delaunay and constrained regular triangulations I: Combinatorial properties. Discrete and Computational Geometry (2008)
[26] Si, H., Gärtner, K.: Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In: Proc. 14th International Meshing Roundtable, San Diego, CA, USA, Sandia National Laboratories, pp. 147–163 (2005)
[27] Bern, M.: Compatible tetrahedralizations. In: Proc. 9th annual ACM Symposium on Computational Geometry, pp. 281–288 (1993)
[28] Liu, A., Baida, M.: How far flipping can go towards 3D conforming/constrained triangulation. In: Proc. 9th International Meshing Roundtable, Sandia National Laboratories, pp. 307–315 (2000)
[29] Karamete, B.K., Beall, M.W., Shephard, M.S.: Triangulation of arbitrary polyhedra to support automatic mesh generators. International Journal for Numerical Methods in Engineering 49, 167–191 (2000)
[30] George, P.L., Borouchaki, H., Saltel, E.: Ultimate robustness in meshing an arbitrary polyhedron. International Journal for Numerical Methods in Engineering 58, 1061–1089 (2003)
[31] Si, H.: TetGen (2007), `http://tetgen.berlios.de`
[32] Möller, T.: A fast triangle-triangle intersection test. Journal of Graphics Tools 2, 25–30 (1997)
[33] Guigue, P., Devillers, O.: Fast and robust triangle-triangle overlap test using orientation predicates. Journal of graphics tools 8, 39–52 (2003)
[34] Hoffmann, C.M.: Geometric and Solid Modeling – An Introduction. Morgan Kaufmann, San Mateo, CA (1989), `http://www.cs.purdue.edu/homes/cmh/distribution/books/geo.html`
[35] Dyken, C., Floater, M.S.: Preferred directions for resolving the non-uniqueness of Delaunay triangulations. Computational Geometry: Theory and Applications 34, 96–101 (2006)