Domain triangulation between convex polytopes

Vasyl Tereshchenko*, Sergiy Pilipenko*, Andriy Fisunenko*

* Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Street, Kyiv, 01601, Ukraine

Abstract

In this paper, we propose a method for solving the problem triangulation of a domain between convex polyhedrons in d-dimensional space using Delaunay triangulation in $O(N^2)$ time. Novelty of our work is in using a modified Delaunay triangulation algorithm with constraints.

Keywords: triangulation, domain, polytope, polyhedron.

1. Introduction

Relevance. This work is concerned with the problem of constructing a tetrahedralization of a domain between a set of convex polytopes in d-dimensional space. A lot of problems of Computational Geometry and Computer Graphics are reduced to this one, e.g. the surface reconstruction from input data captured by satellites which is very important in geo-information systems.

Analysis of previous research results. Goodman’s work raises a question whether the problem could be solved in time $O(N + k^2)$, where $N$ is total number of points and $k$ is number of polyhedrons [1]. As a result of further investigations B. Chazelle founds that the complexity of the solution depends on a form of a domain between polyhedrons [2]. Later B. Joe [10] proposes a new approach which is connected with local transformations of polyhedrons that improves the quality of a triangulation (Delaunay triangulation). However, the performance of this algorithm is $O(N^2 + k^2)$, which is far from desirable.

One of approaches to solve the problem consists of two main stages: construction of the domain and its triangulation. For each stage different methods are used. For example, some methods of constructing domain are based on finding a convex polyhedron around given polyhedrons and then deleting points which belong the given polyhedrons. On the stage of construction of the domain we can use different algorithms of finding a convex hull [3]. To construct a triangulation also there are different ways [4, 5]. In particular, this problem can be solved as a problem of triangulation with constraints [6, 7] or the triangulation problem of non-convex
polyhedron [8]. However, an optimal approach has not been found and the problem remains open.

**Novelty and ideas.** In this paper, we propose a modification of the described above approach, which is based on using a modified Delaunay triangulation algorithm with constraints for domain triangulation.

### 2. Problem and solution method

#### Problem.
To triangulate the domain between \( k \) polytopes which have \( N \) vertices. Every pair of input set of polytopes has the empty domain of intersection.

**Definition 1.** A domain between polyhedrons to triangulate is a domain which is bounded by convex hull of all polyhedrons and their mutually convex facets.

Let us divide problem on two phases: building a domain between polyhedrons and triangulation of a built domain.

**Note 1.** All facets are considered to be triangular. Two adjacent facets may lie in one plane. The input of all algorithms is assumed to be exactly of such kind because any facet can be obviously triangulated onto triangular facets.

Let us consider the method of solving the problem on the example of three-dimensional space.

**Note 2.** Let us introduce a linear order for vertices. Vertex \( v_1 < v_2 \), if and only if \( x \) coordinate of \( v_1 \) is less than \( x \) coordinate of \( v_2 \), or if the \( x \) coordinates of \( v_1 \) and \( v_2 \) are equal and \( y \) coordinate of \( v_1 \) is less than the \( y \) coordinate of \( v_2 \), or if the \( x \) coordinates of \( v_1 \) and \( v_2 \) are equal and \( y \) coordinates of \( v_1 \) and \( v_2 \) are equal and \( z \) coordinate of \( v_1 \) is less than the \( z \) coordinate of \( v_2 \). Onward, let us introduce a linear order procedure for facets. Assume \( v_{11}, v_{12}, v_{13} \) as vertices which define facet \( f_1 \), and \( v_{21}, v_{22}, v_{23} \) vertices that define facet \( f_2 \), moreover, \( v_{11} < v_{12} < v_{13} \) and \( v_{21} < v_{22} < v_{23} \). So, \( f_1 < f_2 \), if and only if \( v_{11} < v_{21} \) or \( v_{11} = v_{21} \) and \( v_{12} < v_{22} \) or \( v_{11} = v_{21} \) and \( v_{12} = v_{22} \) and \( v_{31} < v_{23} \).

#### 2.1. Constructing a domain

For constructing a domain we need to build a convex hull of the set of input polyhedrons. We use "divide and conquer" technique [9]. As a result, we obtain the convex hull of a given set of polyhedrons. The next step is to remove a set of points which belong to the input polyhedra:

1. Form \( F_v \) which is sorted facets of input polyhedrons.
2. Form \( F_m \) which is sorted facets of built convex polyhedron.
3. Form \( F_a \) which is set of facets of domain.
4. Find the smallest facet \( f \) among all facets belonging \( F_v \cup F_m \).
5. If \( f \in F_v \) and \( f \in F_m \), then \( f \) does not belong to the domain, we have to remove it from \( F_v \) and \( F_m \). Else add \( f \) to \( F_a \) and remove \( f \) from \( F_v \) (if \( f \in F_v \)) or \( F_m \) (\( f \in F_m \)).
6. If \( F_v \cup F_m = \emptyset \) then stop, else got to 4.

The result set \( F_a \) will include all facets belonging to the domain, and therefore, the domain has been built.

#### 2.2. Triangulation

Let us consider this problem as the problem of triangulation with constraints.

**Definition 2.** Triangulation of the set of points with constraints in the form of surfaces is a triangulation in which edges of built tetrahedrons do not intersect constraint surfaces (but tetrahedrons’ vertices can lie on constraint surfaces).

To solve the problem of triangulation with constraints let us apply known Hazlewood’s algorithm [6], which consists of four steps: 1) Delaunay Triangulation without constraints; 2) reconstruction of edges of constraints; 3) reconstruction of surfaces of constraints; 4) removing tetrahedra which are outside of the specified domain.
All these steps are described in details in [6]. However, taking into account the specifics of the formulated task the first step has its peculiarities. Therefore, we describe a modified algorithm for constructing the Delaunay triangulation without restrictions related to domain which is bounded by polyhedrons.

2.3. The algorithm for constructing Delaunay triangulation without constraints

Consider domain’s vertices as points which will be used as vertices of triangulation. The triangulation algorithm:

1) Create a set of input vertices \( U \).
2) Create a "superstructure" which is a tetrahedron containing all input vertices.
3) Forming an empty set \( G \).
4) Find a center \( (q) \) and radius of the circumscribed sphere for the superstructure.
5) Select any node of \( U \) and transfer it to \( G \). Delete all tetrahedrons for which \( q \) lies in the circumscribed spheres around them. As a result, a constructed grid forms a polyhedron, which in a general case has a star shape, with any ray that goes from \( q \), must cross the boundary of this polyhedron in a single point. If there are tetrahedrons (adjacent to this polyhedron) for which \( q \) lies in a plane of one of the facets, these facets must also be removed. In fact, edge or facet is removed only in the case of deleting all adjacent tetrahedrons but facets and edges of the superstructure are left untouched. New tetrahedrons are formed by adding edges between \( q \) and vertices of the resulting polyhedron. B. Joe proved that a result will be Delaunay triangulation [3].
6) Find center and radius of the circumscribed spheres for new tetrahedra.
7) If the set \( U \) is not empty, then go to 5, else stop.

As a result we have Delaunay triangulation of vertices of domain, which is located in the tetrahedron.

3. The complexity of the algorithm

**Theorem.** The problem of triangulation of domain can be solved by the proposed method in \( O(N^2) \) time.

**Proof.** Basic steps for constructing a triangulation of a domain between polyhedrons are the following:

1. An algorithm for constructing the convex hull of the set polyhedrons (algorithm "divide and conquer").
2. An algorithm for constructing a triangulation domain (algorithm for Delaunay triangulation with constraints).

In Step 1, we apply the "divide and conquer" technique. Basic steps: searching a starting facet (lower bridge) for the merge procedure and recursive construction of the convex hull [9]. The time complexity of the searching a lower bridge is \( O(N) \) [11] (at each step it is guaranteed that we remove \( N/4 \) vertices):

\[
O(N + (3/4)N + (3/4)^2N + \ldots) = O(N).
\]

In general, the time of constructing the convex hull is \( O(N \log N) \).

In step 2, we apply an algorithm for constructing Delaunay triangulation with restrictions (the complexity is \( O(N^2) \)). Building of the superstructure, inserting one point and number of tetrahedra - \( O(N) \). Inserting all vertices into the triangulation, the reconstruction of all edges and also the reconstruction of constraint surfaces takes time \( O(N^2) \). Checking whether a point lies inside of polyhedron takes time \( O(N) \), hence check of all tetrahedrons will take \( O(N^2) \). Thus, the total complexity of all steps is:

\[
O(N \log N + N \log N + N^2) = O(N^2).
\]

4. Implementation

The practical implementation is made in programming language Java with use of Java3D to visualize data.
For implementation of the algorithm we use "nodes, edges and triangles" data structure [6]. Input of data is performed either by importing xml-file with polyhedrons described as a list of coordinates of vertices and facets or by generation of up to 20 arbitrary polyhedrons with the total number of vertices up to 10 000.

The program provides an ability to observe main stages of the problem solving as shown on Figure 1.

5. Conclusions

Since algorithms which can be generalized to d-dimensional case are used for basic steps of the proposed method it also can be generalized for the case of d-dimensional Euclidean space. The algorithm can be upgraded using different data structures to reduce memory use but it can affect on performance of some of its steps. It is worth noting that the algorithm is very sensitive to the accuracy because auxiliary problems are ill-conditioned (finding center and radius of the circumscribed domains, intersections, etc.). To improve accuracy one can use an exact arithmetics but it can be done by the cost of increasing computational complexity of elementary operations and, hence, the whole algorithm’s performance.

References