An eight-way perturbation technique for the three-dimensional convex hull

Tsuyoshi Ono
Department of Information Science
University of Tokyo, Tokyo 113, Japan
e-mail: ono@is.s.u-tokyo.ac.jp

Abstract

Simulation of Simplicity is a symbolic perturbation technique presented by Edelsbrunner and Mücke [3] to cope with degenerate input data for geometric algorithms. In this paper, we modify some part of this technique and investigate its application to the construction of the convex hull of points in three dimensions. With our technique, a large part of degenerate points on the surface are moved inside, and as the result they are eliminated.

1 Introduction

Three-dimensional convex hulls are geometric objects commonly used to model real-world objects on a computer. Preparata and Hong [4] showed the divide-and-conquer algorithm to find the convex hull of three-dimensional points in optimal time $\Theta(N \log N)$, where N is the number of points. Its implementation in Pascal is presented in detail by Day [1]. A problem yet to be solved in Day's work is the treatment of degenerate input data, which is the key issue in converting theoretical algorithms into working programs. In [1] Day assumes that all of input data is not degenerate (or, *simple*).

Suppose, for example, a subroutine that takes as input a point and a plane (the latter is defined by three points on it) in three dimensions and decides on which side of the plane the point lies. One example of degenerate input is a plane and a point which lies on the plane. If the program should work well with any sort of inputs, the programmer must prepare for every possible degenerate case and often fail to keep their codes simple and readable. Moreover, numerical instability due to round-off errors will be another cause of trouble. It may move degenerate points in an unpredictable direction and often create topological inconsistencies. The program can no longer guarantee its robustness.

Many methods have been proposed to resolve this issue, one of which is a symbolic perturbation. This perturbation labels all input points with indices and alters "the input data in such a way that topological information is unchanged but numerical computations become robust [2]." This enables us to pretend that there is no degenerate input to the program. The programmer would be able to neglect degenerate cases. The perturbation part of code appears only in the procedures of primitive operations. Its drawback, however, is that we must use exact arithmetic, like long integer arithmetic, instead of inexact one, like floating-point arithmetic.

Simulation of Simplicity (SoS) is a symbolic perturbation technique presented by Edelsbrunner and Mücke [3] to cope with degenerate input data for geometric algorithms. The perturbation is conceptual in that it is never computed – it is infinitestimally small due to symbolic variable ϵ. To use SoS, geometric primitives used in algorithms must be written as the sign computation of determinants, namely the orientation of point sequences. The same paper also shows that all operations performed in the convex hull algorithm of Preparata and Hong can be reduced to the computation of determinants. We combined the work of [1] and [3] and realized a prototype version of the implementation of the three-dimensional convex hull algorithm which allows degenerate input data.

Users of symbolic perturbation techniques would like to have some control over the direction of perturbation. For example, an application to test if two polytope have intersection will require the perturbation to move points in an outward direction so that it does not miss any intersection [5, 6].

Here we present another example that needs control of perturbation. Figure 1 illustrates the problem of "surface degeneracy," or a degenerate point on the surface of the resulting hull. The triangle $\triangle ABC$ is a facet of the hull and the point O is inside of the hull. Point D is coplanar with $\triangle ABC$, thus four points A, B, C and D are degenerate.

Suppose we hope that the resulting hull is "clean," that is, we do not want degenerate points remain on the surface. Perturbation schemes, when applied, will cause two possibilities. If point D is perturbed inside, it can be neglected and there is no problem. If perturbed outside, however, D will remain on the surface of the resulting hull, unnecessarily triangulate the facet, and incur extra cost in storage and computation. Moreover, as the dimension rises, the diversity of degenerate cases will also increase and so will the penalty for unnecessary partition in the surface.

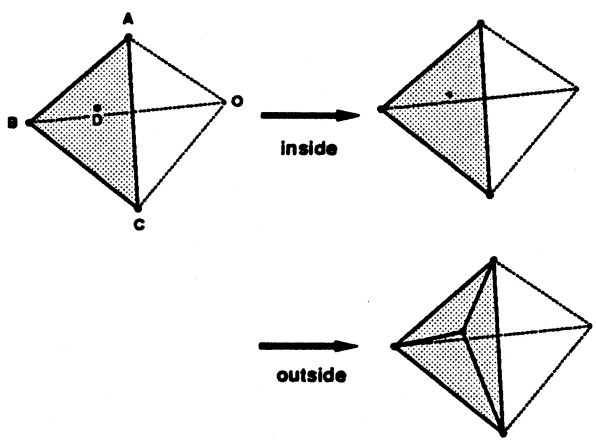

Figure 1: The problem of surface degeneracy

Therefore, It is advantageous to devise a perturbation scheme which perturbs "outward" the vertices of the resulting hull and "inward" most of degenerate points on the surface. (The term "inward" or "outward" is used in a relative context; D is perturbed inside if the amount of outward perturbation of D is smaller than that of A, B or C.)

In this paper, we propose a perturbation technique which avoids a large part of unnecessary triangulation of the surface of the 3-D hull. The technique can be extended to higher dimensions. We observed that a minor repair in the perturbation, not the change in the

algorithm itself, can have the significant gain in computational costs.

In the following section we study the problem of surface degeneracy and propose a minor modification of SoS which successfully eliminates a large part of surface-degenerate points. The key idea is to number input points in such a way that one of three surrounding points (in Figure 1, A, B or C) have a small number and to move it further away from the inside of the hull than other three points, including a surface-degenerate point D.

2 Our Approach

2.1 Preliminaries

Let $P = \{p_0, p_1, \cdots, p_{n-1}\}$ be a set of n geometric objects in d-dimensional space

$$p_i = (\pi_{i,1}, \pi_{i,2}, \cdots, \pi_{i,d}),\ 0 \leq i \leq n-1$$

The orientation of a $d+1$ point sequence $\{p_{i_0}, p_{i_1}, \cdots, p_{i_d}\}$ decides in which side of the halfspace, spanned by $p_{i_0}, p_{i_1}, \cdots, p_{i_{d-1}}$, the query point p_{i_d} lies (we assume $i_0 \leq i_1 \leq \cdots \leq i_d$ without loss of generality). These $d+1$ points are degenerate if all of them lie on the same hyperplane, and otherwise they are non-degenerate, or called simple. Orientation can be decided by a test of a determinant sign of the following matrix.

$$\Lambda_{d+1} = \begin{bmatrix} \pi_{i_0,1} & \pi_{i_0,2} & \cdots & \pi_{i_0,d} & 1 \\ \pi_{i_1,1} & \pi_{i_1,2} & \cdots & \pi_{i_1,d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \pi_{i_d,1} & \pi_{i_d,2} & \cdots & \pi_{i_d,d} & 1 \end{bmatrix}.$$

The sign of $|\Lambda_{d+1}|$ is 0 if $d+1$ points are degenerate, and ± 1 otherwise.

SoS replaces each coordinate by a polynomial in ϵ, a positive infinitesimal variable. Every coordinate $\pi_{i,j}$ is perturbed into

$$\begin{aligned} \pi_{i,j}(\epsilon) &= \pi_{i,j} + \epsilon(i,j) \\ \epsilon(i,j) &= \epsilon^{2^{i \cdot \delta - j}}, \end{aligned} \tag{1}$$

where $\delta > d$. $p_i(\epsilon)$ and $P(\epsilon)$ are the new object and the new set created by this perturbation. They are called ϵ-expansion of the original object p_i and set P.

A symbolic perturbation technique should satisfy two requirements.

1. $P(\epsilon)$ must be simple if $\epsilon > 0$ is sufficiently small.
2. $P(\epsilon)$ must hold all nondegenerate properties of P. For any primitive function PF, $PF(P) = PF(P(\epsilon))$ if P is not degenerate.

The ϵ expansion of Λ_{d+1} is defined as follows:

$$\Lambda_{d+1}(\epsilon) = \begin{bmatrix} \pi_{i_0,1} + \epsilon^{2^{i_0 \cdot \delta - 1}} & \pi_{i_0,2} + \epsilon^{2^{i_0 \cdot \delta - 2}} & \cdots & \pi_{i_0,d} + \epsilon^{2^{i_0 \cdot \delta - d}} & 1 \\ \pi_{i_1,1} + \epsilon^{2^{i_1 \cdot \delta - 1}} & \pi_{i_1,2} + \epsilon^{2^{i_1 \cdot \delta - 2}} & \cdots & \pi_{i_1,d} + \epsilon^{2^{i_1 \cdot \delta - d}} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \pi_{i_d,1} + \epsilon^{2^{i_d \cdot \delta - 1}} & \pi_{i_d,2} + \epsilon^{2^{i_d \cdot \delta - 2}} & \cdots & \pi_{i_d,d} + \epsilon^{2^{i_d \cdot \delta - d}} & 1 \end{bmatrix}.$$

2.2 Eight-way perturbation

We propose a variant of (1):

$$\begin{aligned} \pi_{i,j}(\epsilon) &= \pi_{i,j} + \epsilon(i,j) \\ \epsilon(i,j) &= b\text{-}sign(\pi_{i,j}) \cdot \epsilon^{2^{i \cdot \delta - j}}, \end{aligned} \tag{2}$$

where *b-sign* is a biased sign function such that

$$b\text{-}sign(x) = \begin{cases} +1 & (x \geq 0) \\ -1 & (x < 0) \end{cases}$$

Lemma 2.2.1 *this perturbation meets two requirements mentioned above with regard to the primitive operation that determines the orientation of a sequence of* $d+1$ *points by computing* Λ_{d+1}.

We will show the outline of the proof.

If we rewrite $\Lambda_{d+1}(\epsilon)$ in terms of the subdeterminants of Λ_{d+1}, it becomes the sum of a finite number of terms, each of which is the product of a coefficient, which is a subdeterminant with a possible change in sign, and a so-called ϵ-product, a product of zero or more $\epsilon(i,j)$s.

It is trivial that the second requirement is met if ϵ is small enough. The satisfaction of the first requirement can be proved essentially in the same way as is done in [3]; it is proved from two observations, one is that ϵ-products can not cancel out each other, the other is that at least one term has the coefficient of ± 1. The readers should be referred to [3] for more elaborate explanation.

However, it is worthwhile to mention that ϵ can be small enough (but not zero) to compensate the influence of the coefficients as long as they are constants. Other choices of $\epsilon(i,j)$ could be used, which leads us to other perturbation techniques.

So why choose (2)? The difference from original perturbation of SOS (1) is that the *direction* of perturbation applied to a point varies according to the relative position of the point to the origin. The amount of perturbation added to $\pi_{i,j}$ will be positive if $\pi_{i,j} \geq 0$ and negative otherwise. Intuitively, a point is perturbed as if it 'escapes' from the origin.

Hereinafter, we fix d to 3 and consider only the application of our perturbation technique to three-dimensional convex hull construction. (See Figure 2 for an intuitive illustration of our perturbation technique in three dimensions.) Also we assume that the origin is inside of the resulting hull. Appropriate preprocessing of input data will meet this condition. The direction of perturbation of two points will be same if both are in the same octant and different otherwise. This is an important property of our perturbation technique.

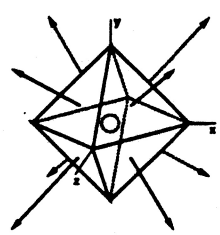

Figure 2: Our perturbation technique in three dimensions

2.3 Behavior of Our Perturbation

This section shows how our perturbation eliminates most of surface-degenerate points in the course of hull construction, with the help of appropriate indexing of input points.

Lemma 2.3.1 *The amount (L_2 distance) of the perturbation experienced by the j-th coordinate of a point labeled with an index i_1 is larger than that of a point labeled with i_2 if and only if $i_1 < i_2$, where i_1, i_2, j are positive integers.*

Proof $|\epsilon(i_1, j)| = \epsilon^{2^{i_1 \cdot \delta - j}} > \epsilon^{2^{i_2 \cdot \delta - j}} = |\epsilon(i_2, j)|$ if $0 < \epsilon < 1$.

This lemma implies that you should label a point with a small index if you want to make it "remains" on the surface.

Suppose that four points q_1, q_2, q_3 and q_4 in the same octant O are coplanar (thus, degenerate points) and q_4 is inside the triangle formed by q_1, q_2, q_3. Also assume the intersection of O and the plane Q defined by q_1, q_2 and q_3 is bounded. This is not a restrictive assumption when we are considering the triangle formed by q_1, q_2 and q_3 as a facet of the resulting hull.

Theorem 1 *If these four points q_1, q_2, q_3 and q_4 are numbered in the decreasing (or increasing) order of z-coordinate and our perturbation (2) is applied, q_4 will be perturbed inside. That is, q_4 and the origin are in the same side of Q.*

Proof q_4 can never be labeled with the smallest number of the four since it is inside of the triangle formed by q_1, q_2 and q_3. Say q_k $(1 \leq k \leq 3)$ is labeled with the smallest number. By lemma 2.3.1, a perturbation experienced by q_k along the z axis is the largest one. Since ϵ can be arbitrarily small, the largest perturbation will obscure other perturbations. q_k will be perturbed in such a way that it escapes from the origin. Thus lemma 1.

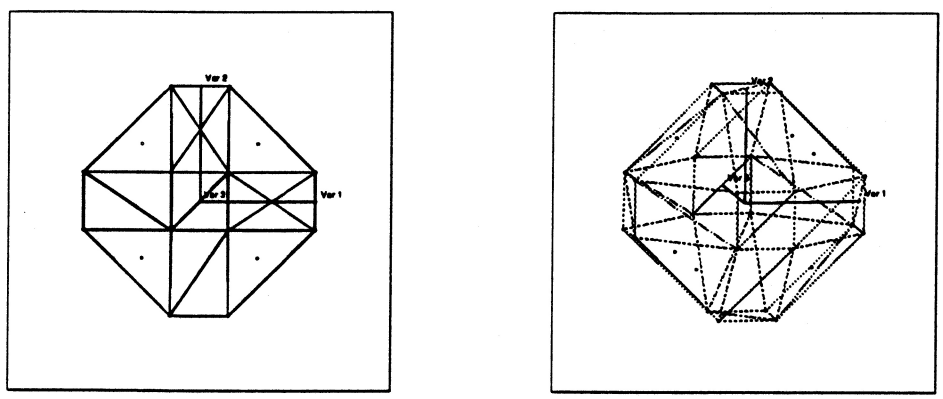

Figure 3: Eight-way perturbation applied to a convex hull

Fig. 3 shows a convex hull with eight degenerate points on its surface. This hull is constructed after input points are processed by our eight-way perturbation technique (2). All of eight surface-degenerate points are perturbed inside and do not triangulate the surface. If SoS is used, four of them will remain on the surface.

3 Discussion

Surface-degenerate points will be or will not be perturbed inside when (1) their surrounding vertices belong to different octants, or (2) the plane coplanar with the surrounding triangle is

not bounded by the octant which the triangle belongs to. We must be careful in the choice of the origin not to increase the number of facets that satisfies condition (1) or (2). If the origin is set too close to the surface, the number will rise. We need a scheme that allows us to position the origin satisfactory near the true center of the hull prior to the hull construction.

As mentioned in the Introduction, a situation may arise in which we wish to have some control over the behavior of perturbations. Our work can be one of the first attempts to meet this need. Apparently, our technique can be used in the applications other than hull construction and can be used in the space of more than three dimensions, as long as the concept of "outward" can be defined. By negating *b-sign*, we get "inward" perturbation, which may be useful if we wish to detect all the surface-degenerate points.

References

[1] Day, A.M. "The implementation of an algorithm to find the convex hull of a set of three-dimensional points," ACM Trans. on Graphics, Vol 9, No 1 (Jan. 1990), pp.105-132.

[2] Day, A.M. "The implementation of a 2D convex hull of algorithm using perturbation, " Computer Graphics Forum, Vol 9 (1990), pp.309-316.

[3] Edelsbrunner,H., and Mücke, E. "Simulation of Simplicity: a technique to cope with degenerate cases in geometric algorithms," Proc.4th Annual Symposium on Computational Geometry (1988).

[4] Preparata, F.P., and Hong, S.J. "Convex hull of a finite set of points in two and three dimensions," Commun. ACM 20, 2 (Feb. 1977), pp.87-93.

[5] Yap, C.K., "A geometric consistency theorem for a symbolic perturbation scheme," Proc.4th Annual ACM Symposium on Computational Geometry (1988), pp.134-142.

[6] Yap, C.K., "Symbolic treatment of geometric degeneracies," Lecture Notes in Control and Information Science, 113, pp.348-358.