Rounding Face Lattices in d Dimensions

[Abstract for Second CCCG, 1990]

Victor Milenkovic
Harvard University
Center for Research in Computing Technology
Cambridge, MA 02138

May 15, 1990

1 Introduction

Recent work in computational geometry has addressed numerical issues that arise in the implementation of geometric programs. The main difficulty is that geometric algorithms require that calculations be carried out to a very high degree of precision to ensure correct behavior of the program. Computing the union of planar polygonal regions whose vertices have k-bit integral coordinates requires $4k$ bits of precision for a standard implementation. The analogous construction in d dimensions[1] requires d^2k bits of precision. Worse, yet, the practical cost of d^2k-bit multiplication is proportional to d^4. Contrast this with the fact that for moderate values of d, the solution to d linear equations in d variables with k-bit coefficients can be carried out using Gaussian elimination with perfect reliability and to very high accuracy using only $2k$-bit "double" precision arithmetic. The requirement for high precision in geometric programs is exacerbated by *cascading*: using the output of one geometric algorithm as the input to another. For example, after m rotations, translations, unions or intersections have been sequentially applied to d-dimensional polyhedral solids, the required precision rises to md^2k bits. The practical cost of this much precision is $O(m^2d^4)$ per multiplication.

Some hope arises in this bleak situation because of the "tantalizing" behavior of geometric algorithms: a well-written $2k$-bit implementation works on almost all inputs and crashes only

[1]That is, the union of solids bounded by simplices whose vertices have k-bit integral coordinates. Such solids arise, for example, as the result of computing the convex hull of a set of points.

occasionally. There are basically two approaches to changing "almost all" to "all" without paying the full cost of standard exact techniques.

Fast Exact Arithmetic: exploit the fact the most computations require only a fraction of the worst case precision.

Robust Geometry: create algorithm which use rounded arithmetic with absolute reliability.

These techniques have been successfully applied in the plane (Fortune, 1989; Milenkovic, 1989; Karasick, Lieber and Nackman, 1989). Unfortunately, cascading ultimately overwhelms fast exact arithmetic. Cascading does not increase the cost of robust algorithms, but as of yet there are no practical robust algorithms for three dimensions. Until such algorithms are developed, we offer here a technique, *high level rounding*, which eliminates the cost of cascading in exact algorithms.

Intersecting two polyhedra with k-bit integral vertices creates vertices with $13k$-bit rational coordinates ($7k$-bit numerator, $6k$-bit denominator). Rotating a polyhedron transforms the vertex coordinates to $5k$-bit rational coordinates.[2] In either case, our strategy will be to round the coordinates to the nearest k-bit integers every time we perform an operation that increases the precision of the coordinates. This technique eliminates the rise in precision caused by cascading and only introduces a small amount of round-off error per operation. However, rounding cannot be done naively: as Figure 1 illustrates, rounding might cause non-intersecting solids (in this case, polygons) to intersect. *High level rounding* refers to any technique that alters the rounding process to avoid introducing intersections. In the case of this paper, we do this by introducing auxiliary vertices in the unrounded objects as illustrated in Figure 2. By introducing vertex **V** on one edge, we create objects whose rounded forms touch but do not overlap.

In the next section we define high level rounding in d dimensions, and in Section 3 we give an algorithm for high level rounding of d-dimensional face lattices.

2 Definitions

Definition 1 (Face Lattice) *A* face lattice *in the plane is a set of line segments in the plane such that any two are disjoint or share a vertex in common.*

A face lattice in three dimensions is a set of triangles such that any two triangles are disjoint share either a single vertex or an entire edge (line segment) in common.

A face lattice in d dimensions is a set of simplices such that any two are disjoint or share an entire sub-simplex of dimension $d' < d$ in common.

[2]Using the quaternion representation of the rotation.

Clearly a polygon or a polyhedron is a special case of a face lattice. In the latter case, we assume that the faces are triangulated.

Definition 2 (Monotonic Rounding Function) *A real function $\rho(x)$ is a* monotonic rounding function *if for all x, x', $x < x'$ implies $\rho(x) \leq \rho(x')$.*

For example, if ρ rounds a real number to the nearest integer, it is a monotonic rounding function.

Definition 3 (ρ-Roundable) *Let ρ be a monotonic rounding function. A face lattice is* ρ-roundable *if it remains a face lattice after the function ρ has been applied to each coordinate of each vertex.*

If we are rounding two polyhedra, it is not permissible for two faces to intersect in their interiors after rounding unless they become identical. However, it is permissible for the two polyhedra to "come in contact" at a vertex, edge or face.

Definition 4 (Lattice Refinement) *Given a face lattice, any face lattice we can arrive at by triangulating some of the simplices is called a* refinement *of the lattice.*

Theorem 1 (Lattice Refinement Theorem) *For any face lattice, there exists a refinement that is ρ-roundable for every monotonic rounding function ρ.*

Note that the refinement is independent of ρ. In the next section, we prove this theorem by giving an algorithm for constructing the refinement.

3 Algorithm

At last year's conference, we described an algorithm that generates a roundable refinement of a two-dimensional lattice. In that case, the refinement depended on ρ; however, it had the nice property that the rounded lattice had exactly as many vertices as the unrefined lattice. See for example Figure 2 where we refine one edge by adding a vertex **V**, but the vertex is merged with another in the rounded lattice. So far, the refinements we can find in higher dimensions increase the number of vertices. The next three paragraphs give algorithms for two, three, and four dimensions. The general algorithm will be given in full paper.

The technique used for refinement is basically a cylindrical decomposition. In two dimensions, pass a vertical line (parallel to the y-axis) through each vertex and add a vertex to each edge where it intersects a vertical line. This process is illustrated in Figure 3. Note that the vertical lines are not added to the lattice. It is easy to prove that this refinement satisfies the theorem.

To create a refinement in three dimensions, project the face lattice into the xy-plane. In the planar projection, add a vertex at the intersection of every pair of intersecting edges. Generate vertical lines and vertices as in the two-dimensional case. Trapezoidalize using vertical line segments. Triangulate each trapezoid by adding a diagonal. Finally, project, in the z-direction, the entire set of triangles onto each of the original triangles. Figure 4 illustrates the projection of a tetrahedron into the plane and subsequent triangulation.

Passing from three to four dimensions is analogous to passing from two to three. Project the four-dimensional lattice into the wxy-space. In that space, generate edges of intersection between any intersecting triangles. Refine the resulting three-dimensional face lattice. Then trapezoidalize by adding faces that span edges which have endpoints sharing wx-coordinates. Tetrahedralize the three-dimensional trapezoids. Project these tetrahedra in the z-direction onto all of the tetrahedra in the original face lattice. Figure 5 illustrates a single three-dimensional trapezoid and its tetrahedralization.

4 Conclusion

The refinement algorithm in the previous section is by no means efficient. In d dimensions, it creates $n^{O(2^d)}$ simplices to refine a face lattice with n simplices; however, the algorithm is not dependent on ρ. There is certainly some saving that can be achieved by allowing this dependency, as there was in the two-dimensional case.

Another objection to this particular type of rounding might be that it "bends" straight or flat objects and otherwise changes the topology of the input. Recent work (Milenkovic and Nackman, 1990), has shown that at least some forms of rounding without bending is NP-hard.

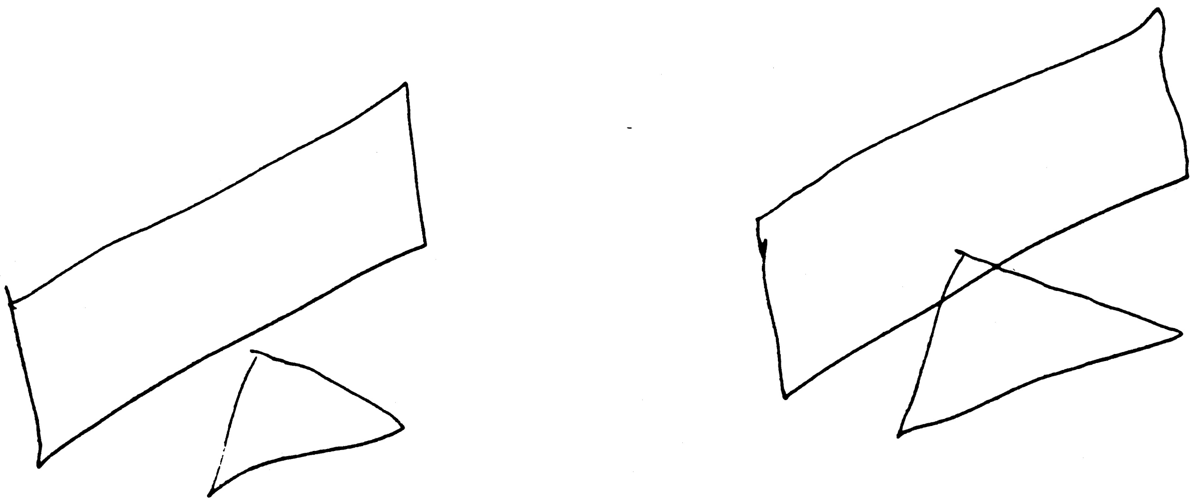

Figure 1: Invalid Rounding

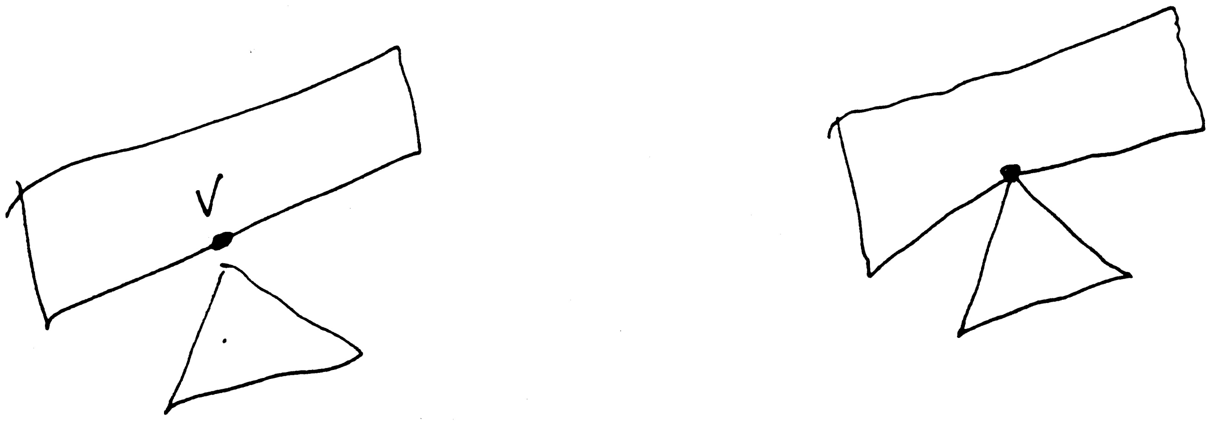

Figure 2: Valid Rounding

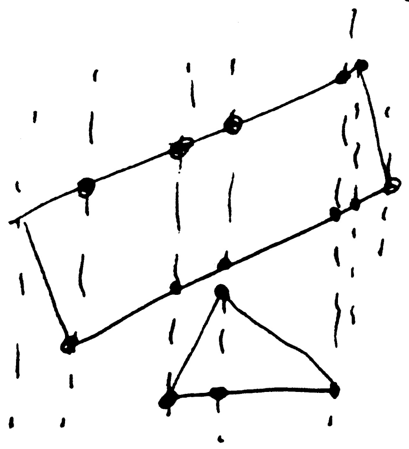

Figure 3: Refinement in Two Dimensions

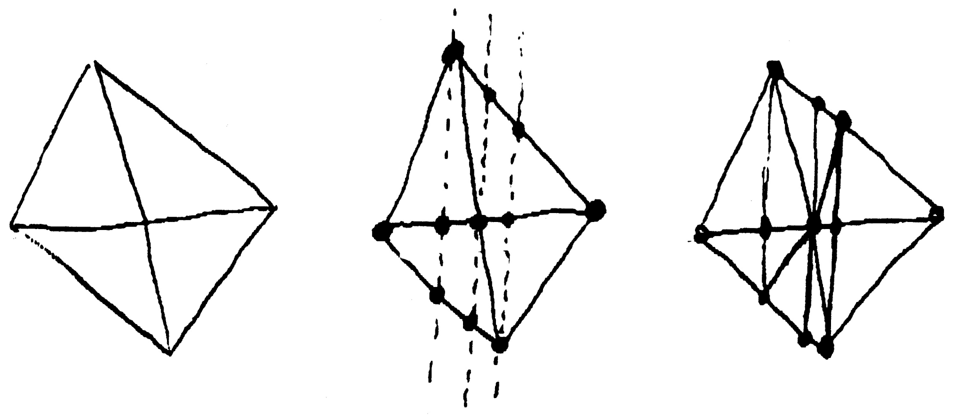
Figure 4: Refinement in Three Dimensions

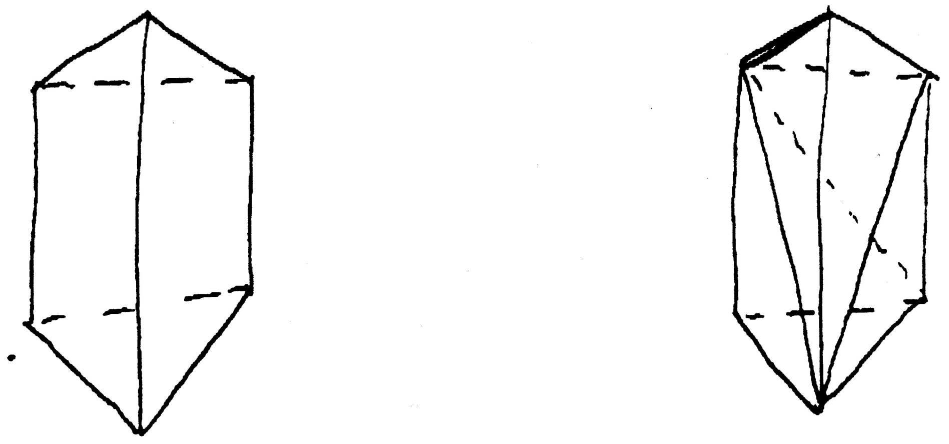
Figure 5: Three-Dimensional Trapezoid and Tetrahedralization