Controlled Perturbation for Arrangements of Polyhedral Surfaces with Application to Swept Volumes*

Sigal Raab†

Abstract

We describe a perturbation scheme to overcome degeneracies and precision problems for algorithms that manipulate polyhedral surfaces using floating point arithmetic. The perturbation algorithm is simple, easy to program and completely removes all degeneracies. We describe a software package that implements it, and report experimental results. The perturbation requires $O(n \log^3 n + nDK^2)$ expected time and $O(n \log n + nK^2)$ working storage, and has $O(n)$ output size, where n is the total number of facets in the surfaces, K is a small constant in the input instances that we have examined, D is a constant greater than K but still small in most inputs, and both might be as large as n in 'pathological' inputs. A tradeoff exists between the magnitude of the perturbation and the efficiency of the computation. Our perturbation package can be used by any application that manipulates polyhedral surfaces and needs robust input, such as solid modeling, manufacturing and robotics. We describe an application for the computation of swept volumes, which uses our perturbation package and is therefore robust and does not need to handle degeneracies. Our work is based on [19] which handles the case of spheres, extending the scheme to the more difficult case of polyhedral surfaces perturbation.

1 Introduction

1.1 Background

There are two major causes for robustness problems in geometric algorithms: the use of floating point arithmetic and degenerate input data. Floating point arithmetic problems are often caused by degenerate data; hence, both problems are closely related. For surveys on robustness in geometric algorithms see, e.g., [12],[17],[21, Chapter 4],[26],[30].

Geometric algorithms are usually designed and proven to be correct in a computational model that assumes exact arithmetic over the real numbers, emphasizing asymptotic time complexity rather than numerical issues. The assumption of exact real arithmetic leads to the assumption of reliable geometric primitives, and this is a viable assumption only in a world with no numerical errors. In reality, most implementations of geometric algorithms are using finite precision floating point arithmetic, where the precision is determined by the hardware of the specific system and varies from one system to the other. Floating point arithmetic is used because of its speed and availability, but a careless use of it might lead to catastrophic errors in practice.

When using floating point arithmetic, a degenerate case is induced not only by degenerate data, but also by close-to-degenerate data. For example, three points do not have to be contained in the same line in order to be considered collinear. It is sufficient that all of them are very close to the same line. Such a degenerate (or almost degenerate) case might lead to wrong answers to predicates and thus wrong code branching. A formal definition of the term *degeneracy* is given in [6]. Most geometric algorithms assume that the input data are in general position (i.e., non-degenerate) and leave the treatment of degenerate cases to the implementor. Often, an application that handles degenerate input is much more complicated than the original algorithm. Also, the treatment of degenerate cases might cause an increase in the algorithm resources, lead to difficult and boring case analysis, and produce cluttered code.

A seemingly straightforward solution to floating point problems is the use of exact arithmetic [2],[4],[5],[8],[39]. Unfortunately, exact arithmetic is often slower compared to floating point arithmetic and needs workarounds for non-rational values (roots, sines, cosines, etc.).

In many cases a computation will be correct even if floating point arithmetic has been used. This leads to floating point filters [14],[33], where all computation steps that give correct results are done by floating point arithmetic, and only those steps that are subject to precision errors are reevaluated by exact arithmetic or tighter approximation. This way, we earn the speed of floating point arithmetic and degrade to slow exact arithmetic only when it is essential.

A paradigm that was devised in order to avoid degenerate cases is *symbolic perturbation* [10],[11],[38]. Perturbations are performed only symbolically by replacing each coordinate of every input geometric object by a polynomial in ε,

*This work is a part of the author's M.Sc. thesis, prepared under the supervision of Dr. Dan Halperin from Tel-Aviv University and Prof. Amihood Amir from Bar-Ilan University. Work on this paper has been supported in part by ESPRIT IV LTR Projects Nos. 21957 (CGAL) and 28155 (GALIA), the Hermann Minkowski–Minerva Center for Geometry at Tel-Aviv University, and the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Grant No. 472/97, D. Halperin).

†Department of Computer Science, Bar Ilan University, Ramat Gan 52900, Israel. Part of this work has been carried out in Tel Aviv University. Email: raab@math.tau.ac.il.

SCG'99 Miami Beach Florida

while maintaining consistency of the input data. Symbolic perturbation requires exact arithmetic.

In a different approach sometimes referred to *as heuristic* epsilons or *epsilon-tweaking* [17],[30],[39], the programmer chooses an arbitrary small value ε. Whenever the absolute difference between two values is less then ε, they are considered equal. Scientific justification is seldom given for choosing any specific ε value, but for the pragmatic user, this approach works well in many cases. Since it is very easy to implement, the user is often willing to absorb the small amount of failures.

We use the name *finite* precision *approximation* for a family of algorithms that perform slight perturbations of the input data [13],[15],[16],[19],[20],[25],[34],[35]. Unlike symbolic perturbation, this is an actual perturbation, namely a perturbation that slightly changes the geometry of the input data. Actual perturbation of the input is justified by the fact that in many geometric problems the numerical input data are real-world data that are either obtained by measuring or modeled in a way that disregards various original properties (e.g., the hard sphere model in [19]). In both cases the computerized model is known to be inaccurate. One approach of this family is *snap* rounding [15],[16],[20], where the data of an arrangement of line segments are slightly changed, so that every vertex is positioned at a center of a pixel and no intersections occur other than in those centers. In [13] snap rounding is extended to three dimensions, with time and place complexity of $O(n^4)$, where n is the total number of vertices, edges, and facets in a polyhedral subdivision. Another of the most recent approaches is described in [19], where degeneracies of spheres are completely removed in a relatively simple scheme. The approach of [19] requires O(n) time, where *n* is the number of input spheres. An incremental procedure is used, where spheres are added one-by-one and if a degeneracy is detected, then only the last sphere that has been added is perturbed. The paradigm presented in [19] is the basis for our work.

1.2 Key Ideas

Motivation Our swept volume application computes the boundary of a collection of three-dimensional polyhedra and employs vertical decomposition [9] as its final step. The vertical decomposition algorithm is very sensitive to robustness problems, and the implementation we use allows no degenerate cases. We apply our robustness package before performing the vertical decomposition, and thus we allow for a successful completion of the program. See Figure 1 for an example of a volume that has degeneracies, created by a triangle swept along a trajectory which intersects itself.

Many geometric applications that need degeneracy-free input (including our swept volume) are using floating point arithmetic, thus our choice of using a finite precision paradigm has obvious advantages: Had we used an exact arithmetic paradigm (e.g., symbolic perturbation), our perturbation scheme would not have been usable to those applications.

Key idea of the Perturbation Scheme Our target users are geometric applications that manipulate polyhedral surfaces; in particular, the swept volume application. Those applications need degeneracy-free input, for instance, a vertex should not touch a non-incident facet. Since we are using floating point arithmetic, we are unable to tell for sure whether a degeneracy exists; we can only tell that a ***potential*** degeneracy exists. For example, we may say that a vertex is potentially touching a facet if it is very close to a facet.

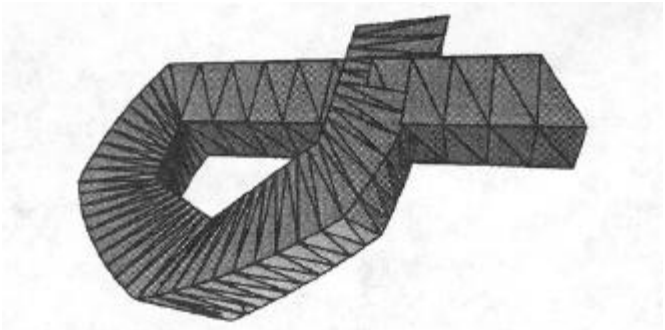

Figure 1: A swept volume of a triangle with degeneracies that arise when its trajectory intersects itself. Our perturbation scheme removes those degeneracies.

In order to define such a potential degeneracy formally, we use a resolution parameter, $\varepsilon > 0$. ε is a small positive real number. Two polyhedral features are assumed too close (and therefore potentially degenerate) whenever they are not e-away from each other (i.e., the distance between two polyhedral features is less than E). The list of potential degeneracies is given in Section 3.2.

We assume ε to be given as an input parameter according to the machine precision. The value of ε depends also on the depth of the expression tree [30]. In our algorithm this tree's depth is a small constant, thus ε is a constant that can be determined and bounded independent of the input size.

Next we define a perturbation radius δ, which depends on ε and on parameters of the input, and is proven to be small enough. A potentially degenerate polyhedral surface is perturbed by at most δ, namely, each vertex of the surface is moved by Euclidean distance of at most δ.

We call our scheme "controlled perturbation". It is controlled in two aspects. First, by determining the size of δ we control the running time of the perturbation scheme and set a tradeoff between the magnitude of the perturbation and the efficiency of the computation. Second, unlike in c-tweaking, our perturbation guarantees that the resulting collection of polyhedral surfaces is degeneracy free.

An obvious limitation of our approach is that we actually change the given placement of the input objects. However, all those changes are small and bounded, and we believe that there are many applications that permit such perturbation, since often their precision is limited to start with (due to measurement limitations, for example).

We extend our perturbation scheme to remove degeneracies which are induced by the vertical decomposition algorithm, used by the swept volume application. In order to remove those degeneracies we perturb the coordinate system and do not change the geometry or topology of the data.

Summary of Results The process completely removes degenerate cases, and requires $O(n \log^3 n + nDK^2)$ time and $O(n \log n + nK^2)$ working storage, and has O(n) output size (see Section 4), where *n* is the number of triangular input facets in all the surfaces together, *K* is a small constant in the input instances that we have examined, ***D*** is a constant greater than *K* but still relatively small in most inputs, and both might be as large as n in 'pathological' inputs (as explained in Section 2). The output size O(n) does not conceal any large constants, and does not depend on ***K*** or ***D.***

Preservation of Topologic Characteristics It is most important to define what characteristics of the polyhedral sur-

face are preserved during the perturbation process. Three approaches are possible: (*i*) Treat the input as a collection of triangles that have no relation to each other (as done in [19] for spheres). In such an approach a perturbation might turn two incident facets into non-incident ones. (*ii*) Treat the input as a collection of distinct (possibly intersecting) polyhedral surfaces, maintain incidence relations of facets, but do not keep intersection data as a part of the data structure. (*iii*) Treat the input as a polyhedral subdivision, which means that even intersection segments (i.e., segments created by the intersection of two non-incident facets) are a part of the data structure and must be maintained during the perturbation process.

We further clarify the difference between approaches (*ii*) and (*iii*) by distinguishing between edges and intersection segments. In approach (*ii*), every edge is a part of the data structure and its incidence relations must be preserved, i.e. its incident facets will not be disconnected from it even if undergoing perturbation. In contrast, a segment is not a part of the data structure, and therefore a segment that might have originally existed in the input data could disappear after perturbation (i.e., a perturbation might turn two intersecting facets into non-intersecting ones). In approach (*iii*) there is no difference between segments and edges: they are both a part of the data structure and incidence relations of both must be maintained.

We have chosen to design our algorithm for approach (*ii*). Approach (*i*) is unacceptable since we target our perturbation scheme for applications that need to keep topologic incidence relations within each surface. Approach (*iii*) preserves more topologic features of the input data, but maintaining intersection relations of the input data leads to a much more complicated algorithm (e.g., must support non-manifold data) and we believe that a large number of geometric applications (such as motion planning and swept volumes) care about the incidence topology of each surface locally, rather than the topology of the full subdivision.

Discussion: Our Scheme vs. Three Dimensional Snap Rounding The 3D snap rounding [13] has the same goal as our perturbation scheme: Both algorithms maintain a robust and error free collection of 3D polyhedral objects, and constitute a finite precision approximation of the input data.

The two algorithms are somewhat difficult for comparison, since we follow approach (*ii*) described above, while [13] follows approach (*iii*). However, since 3D snap rounding is the only previous work which solves a similar problem in finite precision approximation, it is still worth a discussion.

Snap rounding handles subdivisions (i.e., approach (*iii*)) and therefore handles a topologic structure which is more complicated than the one we handle. In particular, it handles non-manifold polyhedral objects, while our scheme does not. On the other hand we believe that our scheme suggests a few solutions to problems that have come up in [13]. While 3D snap rounding is very complicated and seem not to have been implemented, our scheme suggests a fairly simple algorithm which is easy to program (we implement the scheme in a software package and prove its validity in practice). The time and space complexity required by the algorithm in [13] is $O(n^4)$, where our scheme requires only $O(n \log^3 n + nDK^2)$ expected time and $O(n \log n + nK^2)$ working storage, and has $O(n)$ output size (see Section 4). Last but not least, our perturbation scheme totally removes all kinds of degeneracies, while snap rounding removes most degeneracies but encourages one type of degeneracy, namely coincidence of vertices. Snap rounding clearly identifies such degeneracies and therefore the robustness of the output is guaranteed, but the algorithm that uses the output still has to take care of those degeneracies.

2 Preliminaries: Polyhedral Surfaces

Our perturbation scheme manipulates polyhedral surfaces that are manifold, have only triangular facets, and allow intersections (allowing intersections does not contradict the manifold property since there are no incidence relations between the intersecting facets). A polyhedral surface may be arbitrarily large.

We use boundary representation [23] for the polyhedral surfaces and maintain their geometric and topologic information. The geometric information consists of all vertices, where each vertex is represented by three coordinates with constant bit length. The topologic information consists of incidence relations between edges, vertices and facets. E.g., each edge points to incident facets, vertices and edges. The output of our algorithm is the set of perturbed surfaces, using the same data structure with the same bit length.

We base our algorithm on the paradigm presented by Halperin and Shelton [19]. While Halperin and Shelton deal with arrangements of spheres, we deal with arrangements of polyhedral surfaces, which turn out to be more difficult to handle. The reason for the extra complication is that unlike spheres, which can induce degeneracies only because of a relative position of two or more spheres, for polyhedral surfaces we have to remove degeneracies internal to one surface, as well as remove degeneracies induced by relative positions of two or more surfaces. In addition, the structure of a polyhedral surface is more complicated (a polyhedral surface can have arbitrarily many degrees of freedom), thus more degeneracies have to be taken care of.

In a collection of polyhedral surfaces, we assume that the number of facets that a single facet intersects is bounded by a constant K. This assumption is based on a large number of geometric models which we examined, used by the automotive industry and mostly intended for undergoing the swept volume algorithm. Such data describe 3D real world items, which intersect each other only in a controlled and limited manner. Other intersections can be a result of the sweep trajectory in the swept volume application, and there again, the trajectory is planned for pragmatic uses, and thus it does not contain many self intersections. Another constant that is further discussed in Section 4.1 is D. In order to achieve better performance we discretize the three-dimensional space into grid cubes, and D is a bound that is closely related to the number of facets intersecting such a cube. Geometric industrial models do not tend to condense in one zone, and indeed we have found out that D is a constant (greater than K) in the models which we examined. The experimental results supply some findings about K and D. If a user would like to run the perturbation scheme on 'pathological' data (where K or D are not constants and might be as large as n), the perturbation scheme and complexity calculations will still be valid, and the only changes are a bound on δ which depends on n and a greater significance to complexity results that contain K or D.

Let $P = \{P_1, P_2, \ldots, P_m\}$ be a collection of m (possibly intersecting) polyhedral surfaces in $\mathbb{R}^3$. Let $\mathcal{A}(P)$ denote the *arrangement* induced by P, namely, the subdivision of 3-space into cells of dimensions 0,1,2 and 3, induced by the surfaces in P. For a full definition of arrangements see [18],[32]. Notice that while most papers about 3D arrangements of geometric objects assume that each object has constant descriptive complexity (e.g., triangles as in [9]),

we deal with the more complicated form of arrangements of polyhedral surfaces.

3 The Perturbation Scheme

3.1 Notation

Segment denotes the intersection of two non-incident facets.

Intr2 denotes the intersection point of an edge and a non-incident facet.

Intr3 denotes the intersection point of three non-incident facets.

$\boldsymbol{d(p_1, p_2)}$ denotes the Euclidean distance between p_1 and p_2, where p_1, p_2 are three-dimensional points.

$\boldsymbol{d(G_1, G_2)}$ denotes the Euclidean distance between G_1 and G_2 , where G_1, G_2 are three-dimensional geometric entities like edges or facets, and $d(G_1, G_2) = \min\{d(p_1, p_2) \mid p_1 \in G_1, p_2 \in G_2\}$.

$\boldsymbol{B(p, \mu)}$ denotes the ball of radius μ centered at p, namely $\{x \mid d(p, x) \leq \mu\}$, where p and x are three-dimensional points and μ is a positive real number. $B(0, \mu)$ denotes $B((0, 0, 0), \mu)$.

3.2 Inventory of Degeneracies

For a collection of polyhedral surfaces, we define two types of degeneracies. The first type consists of degeneracies inherent to the polyhedral structure (see examples below). The second type consists of degeneracies induced by the using application, and described here for the vertical decomposition algorithm (used by the swept volume application).

Throughout the paper, whenever describing a degeneracy, we use square brackets for terms that would have been used had we been using exact arithmetic.

Inherent degeneracies We will refer to five features of the arrangement: vertex, edge, facet, intersection of two facets (segment), and intersection point of three facets (intr3). A degenerate case is incurred whenever any of the above features is too close to [intersects] any of the other features. The only intersection that is not considered degenerate is when an edge or a segment penetrates the interior of a facet, thus causing an intersection of two or more facets. In [28] we show that many degeneracies can be considered special cases of other degeneracies, and omitting the special cases we end up with four types of inherent degeneracies: (*i*) ***vertex-facet***: A vertex is too close to [touches] a non-incident facet. (*ii*) ***edge-edge***: An edge is too close to [intersects] a non-incident edge. (*iii*) ***edge-segment***: An edge is too close to [intersects] a non-incident segment. (*iv*) ***facet-intr3***: A facet is too close to [contains] a non-incident intr3 [four facets intersect in a point]. We distinguish between two types of inherent degeneracies:

- **Local inherent degeneracies** Inherent degeneracies that occur within one polyhedral surface, e.g., a vertex is too close to [touches] a non-incident facet, both in a single polyhedral surface.
- **Global inherent degeneracies** Inherent degeneracies involving two or more polyhedral surfaces, e.g., a vertex in one surface is too close to [touches] a facet in another surface.

Decomposition induced degeneracies A facet is almost [exactly] vertical, two vertices have x values that are too close [equal], the xy-projection of a vertex and an edge are too close [intersect], the xy-projection of an intr2 and an edge are too close [intersect], or the xy-projection of three edges intersect in three points that are too close [overlap].

3.3 Algorithm Overview

Our perturbation process consists of the following steps:

Local step Remove a subset of the local inherent degeneracies, namely degeneracies not related to intersections. Repeat for every surface in the given collection.

Global step Remove the rest of the local and all of the global inherent degeneracies.

Coordinate system step Remove decomposition-induced degeneracies. Do that by perturbing the coordinate system orientation.

We would like the perturbation scheme to be as simple as possible. Simplicity is achieved when the topology and the internal relative geometry of the polyhedral surfaces are not changed. Choosing big perturbation units (e.g., a whole polyhedral surface) will achieve the simplicity goal, but will not remove internal inherent degeneracies. In order to solve this conflict we chose to remove inherent degeneracies in two steps. The local step perturbs vertices inside a single polyhedral surface, and therefore does not keep the simplicity goal. Thus we try to minimize the local step and use it to remove only degeneracies that cannot be removed by the global step. The global step uses large perturbation units (later on we will see that those units are terrains) and keeps the simplicity goal. By the end of the global step all the inherent degeneracies are removed.

3.4 Local Step

In this step our perturbation unit will be one vertex in a polyhedral surface. We remove only internal inherent degeneracies that cannot be removed by the global step, namely *vertex-facet* and *edge-edge*.

We remove the degeneracies in each polyhedral surface locally, by an incremental procedure where we add the vertices of each surface one by one and if a degeneracy is detected we only perturb the last vertex that has been added.

Let $P = \{P_1, P_2, \ldots, P_m\}$ be a collection of m (possibly intersecting) polyhedral surfaces. Let s_i be the number of vertices in P_i, $1 \leq i \leq m$. Let $v_1, v_2, \ldots, v_{s_i}$ be an ordering of the vertices in P_i. Let Q_r denote the data structure that was generated while processing vertices $1, \ldots, r$, $1 \leq r \leq s_i$. Q_r contains the possibly perturbed vertices $v_1, v_2, \ldots, v_r$, and all the edges and facets of the current polyhedral surface P_i, whose incident vertices are in $v_1, v_2, \ldots, v_r$.

Let δ_1 denote the maximum perturbation radius of the local step. After the completion of stage r for P_i, the following invariants are guaranteed by the incremental procedure: (*i*) Any vertex in Q_r has been moved by Euclidean distance of at most δ_1 from its original placement. (*ii*) $d(v, f) > \varepsilon$ for every pair of non-incident vertex v and facet f in Q_r. (*iii*) $d(e_1, e_2) > \varepsilon$ for every pair of distinct non-incident edges e_1, e_2 in Q_r.

A perturbation of a vertex v_r is done by choosing its new location v'_r uniformly at random within the ball $B(v_r, \delta_1)$. Invariants (*ii*) and (*iii*) define forbidden loci F_2 and F_3 for v'_r respectively. Let $E_r := B(v_r, \delta_1) \setminus (F_2 \cup F_3)$. We keep choosing v'_r inside $B(v_r, \delta_1)$ until the chosen location is inside E_r. In Section 3.6 we describe how we fix δ_1 and explain why this choice leads to a valid location (i.e., inside E_r) with high probability. Suppose the procedure has been

completed successfully for the first $r-1$ stages. Placing v'_r inside $B(v_r, \delta_1)$ guarantees invariant (i). Placing v'_r outside the forbidden loci F_2 and F_3 guarantees invariants (ii) and (iii). Notice that if v_r already keeps the invariants, no perturbation will take place. The region F_3 is a union of sphere slices and is described in the Appendix. Both F_2 and F_3 are described in [28].

3.5 Global Step

In this step we remove global inherent degeneracies and local inherent degeneracies that have not been removed in the local step. The global step proceeds in three sub-steps:

xy-mono partitioning Partition each polyhedral surface into terrains (i.e., xy-monotone surfaces). Define the perturbation units to be terrains. A polyhedral terrain that has undergone the local step is free of **any** local inherent degeneracies, thus each perturbation unit is now degeneracy free.

perturbation Perturb each terrain as a rigid object in order to remove global inherent degeneracies.

stitching Stitch formerly incident terrains by creating *connectors* between them. See Figure 2.

Partitioning into xy-Monotone Surfaces A surface is xy-monotone if every line orthogonal to the xy-plane intersects the surface in at most one point. It is guaranteed that no local inherent degeneracy exists in a terrain: Degeneracies of type *vertex-facet* and *edge-edge* have been removed during the local step, and all other local degeneracies can exist only when there are self intersections, which do not occur in an xy-monotone surface. Hence, a local degeneracy of type *edge-segment* or *facet-intr3* that has previously existed in a polyhedral surface, is now becoming a global degeneracy after the surface has been partitioned into terrains.

Before performing the partitioning we choose a coordinate system so that no facet is vertical, using a technique that is similar to the one in the coordinate system step. In our software package we use a fairly straightforward partitioning algorithm which will not be discussed here.

Perturbation In the global step we set the perturbation unit to be a whole polyhedral terrain. This ensures that the topology and the internal relative geometry of each terrain is preserved. In addition we make sure that the perturbation is done only by translation and not by rotation, which ensures that the orientation of each terrain is preserved.

We remove the global degeneracies by an incremental procedure where we add the polyhedral terrains one by one and if a degeneracy is detected we only perturb the last terrain that has been added. In addition to testing the validity of the (possibly) perturbed terrain, we make sure that no degeneracies are induced by the new connectors which stitch that terrain to incident terrains that were added previously.

Let $T = \{T_1, \ldots, T_l\}$ be an ordering of the polyhedral terrains that have been created out of all the input polyhedral surfaces. Once a terrain T_j, $1 \leq j \leq l$, has been processed, let $con(T_j)$ denote the set of connectors created when the possibly perturbed T_j is stitched to formerly incident terrains T_k, $k < j$. Let M_j denote the data structure that was generated while processing terrains $1, \ldots, j$, $1 \leq j \leq l$. M_j is a collection of polyhedral surfaces comprised of the possibly perturbed $T_1, \ldots, T_j$ and the connectors $con(T_1), \ldots, con(Tj)$.

Let δ_2 denote the maximum perturbation radius of the global step. Unlike the local step, here we do not choose a new location for one vertex; instead we choose a translation vector for the whole terrain. Again we choose a point uniformly at random, but this time inside the ball $B(0, \delta_2)$. The chosen point defines the translation vector for the terrain.

After the completion of stage j the incremental procedure guarantees that any of the terrains $T_1, \ldots, T_j$ has been moved by at most δ_2, and that there are no inherent degeneracies in M_j. In detail, after the completion of stage j, the following invariants are guaranteed: (i) Any terrain in M_j has been moved by Euclidean distance of at most δ_2 from its original placement. (ii) $d(v, f) > \varepsilon$ for every pair of non-incident vertex v and facet f in M_j. (iii) $d(e_1, e_2) > \varepsilon$ for every pair of distinct non-incident edges e_1, e_2 in M_j. (iv) $d(e, s) > \varepsilon$ for every pair of non-incident edge e and segment s in M_j. (v) $d(f, i) > \varepsilon$ for every pair of non-incident facet f and intr3 i in M_j.

The invariants $(ii), \ldots, (v)$ define forbidden loci $F_2, ..., F_5$ respectively. Let $E_j := B(0, \delta_2) \setminus (F_2 \cup \ldots \cup F_5)$. We keep choosing random points inside $B(0, \delta_2)$ until the chosen translation vector is inside E_j. Section 3.6 supplies more details about δ_2 and explains why the choice of δ_2 leads to a valid perturbation with high probability. Suppose the incremental procedure has been carried out successfully for the first $j-1$ stages. Choosing the translation vector inside $B(0, \delta_2)$ guarantees invariant (i). Choosing the translation vector outside the forbidden loci $F_2, \ldots, F_5$ guarantees invariants $(ii), \ldots, (v)$. Notice that if T_j already keeps the invariants, no perturbation will take place. The region F_3 is a union of Minkowski sums[1] of a 2D parallelogram and a ball $B(0, \varepsilon)$, and is described in the Appendix. All forbidden loci $F_2, \ldots, F_5$ are described in [27].

Stitching A *connector* is created in the following way, depicted in Figure 2(c): An edge is created between each pair of partitioned vertices, inducing a parallelogram for every partitioned edge. The parallelogram is split into two triangles by adding a diagonal.

Connectors are created after each stage of the incremental procedure, in order to stitch the terrain that has been possibly perturbed in that stage to former incident terrains. If two originally incident terrains are not perturbed, then no connector will be created and their original adjacency relations will be restored (the terrains will be 'glued' back).

The creation of connectors restores the connectivity of polyhedral surfaces that were partitioned into terrains. Since a perturbation is done in tiny distances (δ_2 is very small), a connector will be very narrow, and hardly noticed compared to a regular facet. It is guaranteed that the connectors do not induce any new degeneracies, by additional tests that we carry out when a terrain is being perturbed. In order to test the connectors validity, we treat them as independent polyhedral surfaces, compute their intersection with other terrains and other connectors, and test them for degeneracies as we do for terrains. The connectors contribute forbidden loci that affect the value of δ_2. Analyzing those loci is very complicated and explained in detail in [27], but the implementation remains simple.

Notice that the stitching procedure works well as long as at most two terrains meet in one point. In a large family of input data this is guaranteed. In cases where more than two terrains meet at a point, we run an additional stitching procedure, which is straightforward and will not be described here. The additional procedure works well in practice but we do not give a guaranteed bound on the running time as

[1] The Minkowski sum of two sets of points S_1 and S_2, denoted $S_1 \oplus S_2$, is defined as $S_1 \oplus S_2 := \{p + q \mid p \in S_1, q \in S_2\}$.

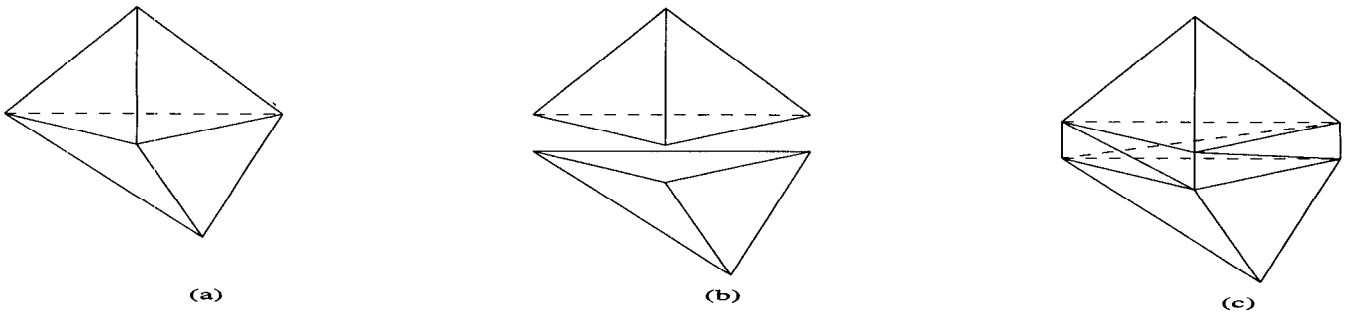

Figure 2: (a) Original polyhedral surface. (b) Partitioning into xy-monotone terrains and perturbing one of them. (c) Creating connectors (in reality the connectors are much thinner; here they are widened for clarity)

a function of the perturbation size in this case; we leave the δ bound proofs for future research.

It is evident that the addition of connectors changes the original structure of the input polyhedral surfaces. However we see that as a minimal harmless change in the context of our application: the connectors are very narrow strips within the input polyhedral surfaces, and the overall geometric shape of the surface is maintained.

3.6 Choosing δ for the Local and Global Steps

In both the local and global steps, we wish to choose a point inside a ball, avoiding forbidden regions of the ball. Let B be the ball and let F be the union of the forbidden volumes. Let $E := B \setminus F$ denote the valid placements. We are interested in choosing a point inside E.

The way of selecting a point inside E is by choosing a point uniformly at random inside B. We would like to have probability of more than $\frac{1}{2}$ for choosing a valid placement for that point, and this is guaranteed once we have $Volume(B) > 2 \cdot Volume(F)$. Our calculations (see Appendix) show that $Volume(F)$ is always finite and that we need to choose $\delta_1 = 6.3\,\varepsilon^{1/6}K^{1/3}LV^{1/3}$ and $\delta_2 = 4.8\,\varepsilon^{1/6}KLV^{1/3}$, where K is the maximum number of facets in P intersecting any single facet, L is the maximum edge length, V is the maximum number of vertices in one surface, and assuming $K \geq 10$, $V \geq 10$, $L \geq 10$, $\delta_1 \leq L$, $\delta_2 \leq L$ and $\varepsilon \leq 10^{-4}$. Notice that neither δ_1 nor δ_2 depend on the value of D.

Let $\delta := \delta_1 + \delta_2$. Any vertex in P was moved by Euclidean distance of at most δ. Our experimental results show that the bound on δ given above is very conservative. This is not surprising, since δ is a theoretical worst-case bound, and even as such it shows that our approach does not conceal any very large constant.

3.7 Coordinate System Step

In this step degeneracies induced by the vertical decomposition algorithm are removed. We remove the degeneracies by a perturbation of the coordinate system orientation (i.e., no geometric or topologic changes to the data are incurred). The technique used is similar to the perturbation of the z-axis in [19], i.e., choosing an orientation uniformly at random and making sure that no decomposition-induced degeneracy is detected. For lack of space we postpone further details to the full paper.

Notice that the techniques in this step have been chosen due to the nature of the decomposition-induced degeneracies. Had we been using the perturbation scheme for a different application (e.g., an application where three collinear points are considered degenerate), then we would have possibly used a different perturbation technique.

4 Complexity Analysis

4..1 Time Complexity

Let L be the maximum length of an edge in the input data. We discretize the 3D space into grid cubes of edge length $L+2\varepsilon$ each. We use the term *expanded entity* for the Minkowski sum of an entity and the ball $B(0,\varepsilon)$. An expanded facet can intersect at most 8 grid cubes (i.e., in a cube whose edge length is double the size of a grid cube edge).

For each grid cube we maintain a list of the expanded entities intersecting that cube. We assume that the number of expanded facets intersecting a single cube is bounded by a constant D, as explained in Section 2. By the definitions of K and D we get that the number of vertices and edges in one grid cube is bounded by $3D$, and that the number of intr3 entities in one grid cube is bounded by $D\binom{K}{2}$.

Local Step For removing degeneracies of type *vertex-facet*, we look at all the cubes that an expanded facet intersects (at most 8) and check for intersection with the vertices in those cubes (at most $3D$ per cube). Since there are n facets, we get that at most $24nD$ combinations of a vertex and a facet need to be checked. In a similar way, for removing degeneracies of type *edge-edge*, we infer that at most $72nD$ combinations of two edges need to be checked. This, together with the choice of δ_1, guarantees expected time of $O(nD)$.

Global Step The complexity of the partitioning sub-step is $O(n\log^3 n)$: We apply a breadth first search over the facets which have not been assigned to terrains yet. Each facet f visited during the search can contribute two new edges e_1, e_2 and one new vertex v to the currently created terrain T. The facet fits the terrain if the following are true (where $\bar{o}$ denotes the projection of o on the xy-plane): (i) $\bar{v}$ is not inside the incident facet in $\bar{T}$ $(O(1))$. (ii) The whole terrain $\bar{T}$ is not inside $\bar{f}$ $(O(1))$. (iii) $\bar{e}_1, \bar{e}_2$ do not intersect any of the edges of $\bar{T}$ $(O(\log^3 n)$ by [7]).

During the perturbation sub-step the operation that has the highest complexity is the one where we check for degeneracies of the type *facet-intr3*, where we look at all the cubes that an expanded facet intersects (at most 8) and check for intersection with all the intr3 in those cubes (at most $D\binom{K}{2}$ per cube). This is done for each facet in the input data, so we get that at most $8nD\binom{K}{2}$ combinations of a facet and an intr3 need to be checked, which is $O(nDK^2)$. In summary, together with the choice of δ_2, the guaranteed expected time of the global step is $O(n\log^3 n + nDK^2)$.

Remark: Our algorithm has finite expected running time and it gives a correct answer when it stops. Therefore it can

be transformed into a Las Vegas algorithm with the same expected asymptotic running time.

Coordinate System Step The operation that has the highest complexity in this step is the detection of three edges whose xy-projections intersect in three points that are too close [overlap]. Since every triple of edges has to be checked, we get a time complexity of $O(n^3)$. It is important to emphasize that the coordinate system step removes degeneracies that are not inherent, thus considered exterior to the perturbation scheme in two aspects: First, many geometric applications can use our perturbation scheme, and most of them do not need vertical decomposition and hence do not need to remove decomposition-induced degeneracies. Second, related research about robustness of three-dimensional polyhedral surfaces focuses on inherent degeneracies only. Therefore, we do not consider this step as an integral part of our scheme and we thus do not include its time requirements in our complexity summary.

4.2 Space Complexity

Output Size The only increase in the space of the input data is while creating the connectors. Each connector adds 2 facets to the polyhedral surface and there can be at most $O(n)$ connectors. Therefore we have an output size of $O(n)$. Notice that this size is independent of K and D.

Working Storage We need $O(n \log n)$ space for the substep of partitioning into xy-monotone terrains (for the procedure described in 4.1; see [7]), and $O(nK^2)$ space for maintaining lists of intersection entities. Altogether we obtain a working space of $O(n \log n + nK^2)$.

Theorem 4.1 *Given a collection P of polyhedral surfaces with a total number of n triangular facets, and a resolution parameter $\varepsilon > 0$, a valid perturbation of the surfaces in P can be computed in $O(n \log^3 n + nDK^2)$ expected time and $O(n \log n + nK^2)$ working storage, and has $O(n)$ output size, where K is the maximum number of facets intersecting any single facet in P, L is the maximum edge length in P, and D is the maximum number of expanded facets intersecting a grid cube of edge length $L + 2\varepsilon$. The perturbation is obtained by moving each vertex by Euclidean distance of at most δ from its original placement so that all the inherent degeneracies are resolved, where δ is a parameter that depends on ε, K, L, and the maximum number of vertices in one polyhedral surface. In expected $O(n^3)$ time we can also find a rotation of the coordinate system so that all the decomposition-induced degeneracies are removed.*

5 Swept Volume

A *swept volume* is defined as the geometric space occupied by an object moving along a trajectory in a given time interval. The motion can be translational and rotational. The moving object, which can be a surface or a solid, is called a *generator*. The motion of the generator is called a *sweep*.

Swept volumes play an important role in many geometric applications, such as geometric modeling, robot motion planning, numerical control cutter path generation, and assembly planning. For related research about swept volumes see, e.g., [1],[3],[22],[29],[31],[36],[37].

For our swept volume application we have chosen to implement the algorithm given by Abrams and Allen [1], which matches our needs. Namely, it applies volume sweeping of a polyhedral body, moving along a general trajectory in a motion that can be translational and rotational as well, and outputs a faceted approximation of the result. The biggest drawback reported in [1] is the robustness problem, where the arrangement computation fails due to floating point errors and degenerate cases. We have solved the problem by using our perturbation package. We choose to perform the arrangement computation by the vertical decomposition algorithm described in [9], extended to the case of polyhedral surfaces. This algorithm requires degeneracy-free input, hence before using it we apply our perturbation package.

n	K	k	d	L	l	V	time
62	6	0.8	11.8	1794	925.4	33	1.01
302	9	0.4	22.8	909.9	625.5	153	9.35
1202	9	0.3	73.3	900.6	588.5	603	123.49

Table 1: Time (in seconds) to remove inherent degeneracies. The objects are swept volumes of the same generator moving along the same trajectory, in different refinement levels. Changes in K, $\bar{k}$, $\bar{d}$, L, $\bar{l}$, and V are shown too.

6 Experimental Results

Three more variables besides n, K, D, L, V affect the results. In a given collection of polyhedral surfaces, let k be the number of facets intersecting a given facet, d the number of expanded facets intersecting a given grid cube, and l the length of a given edge. Thus K, D and L are the maximum of k, d and l. Let $\bar{k}$, $\bar{d}$ and $\bar{l}$ denote the average values of k, d and l respectively. Timing results are affected more by the average values than by the maximum values, and therefore we supply them in some of the tables as well. All timings were done on a Pentium II 450MHz and 512MB RAM, under Linux. The objects tested in Tables 1, 2 and 4 can be found in [28]. Many of the software classes used in our implementation belong to the CGAL[2] library.

In Table 1 we give time results for several swept volumes, which differ in the number of input facets. In the results we specify the time needed to remove inherent degeneracies. In order to supply a basis for comparison, we use ε=1e-8 and δ=1e-4 for all the experiments, and choose all examples to be the swept volume of the same generator moving along the same trajectory. Our swept volume application enables a trajectory refining mechanism, and by using different refinement levels we obtain a different number of facets for each object. Notice that K,$\bar{k}$,L,$\bar{l}$, and V are also affected by the refinement level. This table also illustrates that $\bar{d}$ is affected by the density of the facets rather than by their number: the refinement process increases the density of the facets significantly, and indeed its affect on $\bar{d}$ is striking. The swept volume of 302 facets is depicted in Figure 1.

In Table 2 we show the tradeoff between the magnitude of the perturbation and the efficiency of the computation, introduced by varying values of δ and ε. The perturbation procedure is run on a swept volume of 842 facets, with ε that varies in the range $[1e-3, 1e-8]$ and δ that varies in the range $[1e-1, 1e-6]$. This table illustrates that the theoretical bounds obtained for δ in Section 3.6 are crude and that in practice these bounds are much smaller.

A large portion of the research concerning 3D arrangements deals with triangles. In Table 3 we test arrangements of random triangles, where each triangle has a random origin limited to a cube of a given size, and a random orientation.

[2]CGAL: Computational Geometry Algorithms Library, see http://www.cs.uu.nl/CGAL

$\delta\backslash\varepsilon$	1e-3	1e-4	1e-5	1e-6	1e-7	1e-8
1e-1	48.03	45.48	42.47	41.11	41.08	39.94
1e-2		48.06	46.29	43.96	43.30	40.33
1e-3			48.14	45.85	43.56	43.04
1e-4				46.36	44.03	43.97
1e-5					52.34	44.42
1e-6						48.41

Table 2: Time (in seconds) to remove inherent degeneracies, introducing the tradeoff between the magnitude of the perturbation and the efficiency of the computation. The examined object is of size 842 with $\bar{k}$ of 0.4 and $\bar{d}$ of 31.3.

	dense: 10		medium: 30		sparse: 80	
	K	time	K	time	K	time
uniform size: 5	50	401.5	5	14.7	1	3.9
2 sizes: 2,8	63	552.1	8	18.9	2	4.0
random size: [0-10)	77	405.4	9	17.8	2	4.2

Table 3: Time (in seconds) to remove inherent degeneracies and K values, for a collection of 500 random triangles. Triangles edge lengths are specified in the leftmost column. Cube edge lengths (bounding triangles origins) are specified in the uppermost row.

The cube size affects the density of the triangles, and we compare running time and the intersection parameter K for different density levels. We have chosen cube sizes of 10, 30 and 80 in order to achieve dense, medium dense and sparse inputs respectively. For all inputs we use triangles with average edge length of 5, and compare results for different groups of edge lengths, namely uniform length of 5, two different lengths (2 and 8) and random lengths in the range [0-10).

In Table 4 we compare the time ratio for removal of inherent degeneracies out of swept volumes that have some extreme characteristics. The examined ratios are the ratio of time spent during the local step, global step and during connectors manipulation, out of the whole total time of inherent degeneracies removal. (Notice that connectors manipulation is a part of the global step). The first volume has no degeneracies at all, the second one (depicted in Figure 3) has a dense intersection area that lowers the chances of finding a valid perturbation, the third one has only local degeneracies, and the fourth one has degeneracies that induce a large number of connectors. We use ε=1e-8 and δ=1e-3 for all the experiments, and choose all of the examined volume sizes to be between 916 to 932 facets. Although the sizes of the four models are almost the same, their $\bar{d}$ values are significantly different, as their densities are different.

Acknowledgements

The author wishes to thank Dan Halperin for supervising the research and contributing useful ideas. The author also wishes to thank Iddo Hanniel, Sariel Har-Peled, Oren Nechushtan, Lutz Kettner and Hayim Shaul for helpful discussions concerning the problems studied in this paper. Lutz Kettner has also written the Polyhedron software classes [23] which have been used during the experiments.

References

[1] S.A. Abrams and P.K Allen. Swept volumes and their use in viewpoint computation in robot work-cells. In *Proc. IEEE Intl. Sympos. on Assembly and Task Planning*, pages 188–193, 1995.

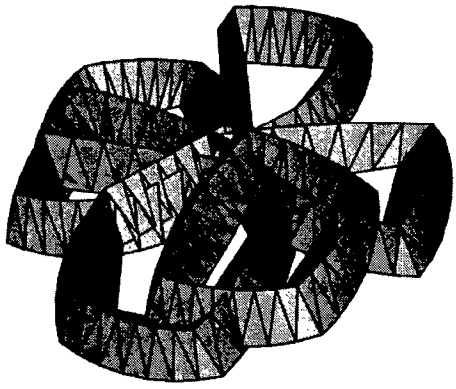

Figure 3: A swept volume inducing many degeneracies. The dense intersection area in the center makes it difficult to find a valid perturbation.

file name	$\bar{k}$	$\bar{d}$	total	$\frac{local}{total}$	$\frac{global}{total}$	$\frac{con}{total}$
no_deg	0	17.5	17.1	0.2	0.8	0.003
degenerate	2.5	26.9	76.7	0.09	0.91	0.08
local_deg	0.3	58.1	73.6	0.23	0.77	0.002
many_con	1.7	6.5	393.4	0.01	0.99	0.26

Table 4: Time ratio for removal of inherent degeneracies out of swept volumes that have some extreme characteristics. The total time is given in seconds. All tested objects have similar size of about 925 triangles.

[2] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparat a, and M. Yvinec. Evaluation of a new method to compute signs of determinants. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C16–C17, 1995.

[3] D. Blackmore and M.C. Leu. Analysis of swept volume via Lie groups and differential equations. *The Intl. J. of Robotics Research*, 11(6):516–536, 1992.

[4] H. Brönnimann, I. Emiris, V. Pan, and S. Pion. Computing exact geometric predicates using modular arithmetic with single precision. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 174–182, 1997.

[5] H. Brönnimann and M. Yvinec. Efficient exact evaluation of signs of determinants. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 166–173, 1997.

[6] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 16–23, 1994.

[7] Y.J. Chiang, F.P. Preparata, and R. Tamassia. A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. *SIAM J. Comput.*, 25:207–233, 1996.

[8] K. L. Clarkson. Safe and effective determinant evaluation. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 387–395, 1992.

[9] M. de Berg, Leonidas J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. *Discrete Comput. Geom.*, 15:35–61, 1996.

[10] H. Edelsbrunner and E. Mücke. Simulation of simplicity: A technique to cope degenerate cases in geometric algorithms. *ACM Trans. on Graph.*, 9(1):66–104, 1990.

[11] I.Z. Emiris, J. Canny, and R. Seidel. Efficient perturbations for handling geometric degeneracies. *Algorithmica*, 19(1-2):219–242, September 1997.

[12] S. Fortune. Robustness issues in geometric algorithms. In *[24]*, pages 9–14.
[13] S. Fortune. Vertex-rounding a three dimensional polyhedral subdivision. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 116–125, 1998.
[14] S. Fortune and C. J. van Wyk. Static analysis yields efficient exact integer arithmetic for computational geometry. *ACM Trans. Graph.*, 15(3):223–248, July 1996.
[15] M. Goodrich, L. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.
[16] D. Greene and F. Yao. Finite resolution computational geometry. In *Proc. 27th IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.
[17] L.J. Guibas. Implementing geometric algorithms robustly. In *[24]*, pages 15–22.
[18] D. Halperin. Arrangements. In J.E. Goodman and J. O'Rourke, editors, *CRC Handbook of Discrete and Computational Geometry*, chapter 21, pages 389–412. CRC Press, Inc., Boca Raton, FL, 1997.
[19] D. Halperin and C.R. Shelton. A pertubation scheme for spherical arrangements with application to molecular modeling. *Comput. Geom. Theory Appl.*, 10:273–287, 1998.
[20] J. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.* To appear.
[21] C. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, California, 1989.
[22] Z.J. Hu and Z.K. Ling. Swept volumes generated by natural quadric surfaces. *Computers & Graphics*, 20(2):263–274, 1996.
[23] L. Kettner. Designing a data structure for polyheral surfaces. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 146–154, 1998.
[24] M.C. Lin and D. Manocha, editors. *Applied Computational Geometry (Proc. WACG '96)*, volume 1148 of *Lecture Notes Comput. Sci.* Springer, 1996.
[25] V.J. Milenkovic. Verifiable implementation of geometric algorithms using finite precision arighmetic. *Artificial Intelligence*, 37:377–401, 1988.
[26] F.P. Preparata. Robustness in geometric algorithms. In *[24]*, pages 23–24.
[27] S. Raab. Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes. M.Sc. thesis, Dept. Comput. Sci., Bar Ilan University, Ramat Gan, Israel. Forthcoming.
[28] S. Raab. Excerpts from M.Sc thesis [27], in http://www.math.tau.ac.il/~raab/thesis_cites.html. 1998.
[29] K. Sambandan, K. Kedem, and K.K. Wang. Generalized planar sweeping of polygons. *J. of Manufacturing Systems*, 11(4):246–257, 1992.
[30] S. Schirra. Precision and robustness in geometric computations. In M. van Kreveld, J. Nievergelt, T. Roos, and P. Widmayer, editors, *Algorithmic Foundations of Geographic Info. Syst.*, volume 1340 of *Lecture Notes Comput. Sci.*, chapter 9, pages 255–287. Springer, 1997.
[31] W.J Schroeder, W.E. Lorensen, and S. Linthicum. Implicit modeling of swept surfaces and volumes. In *Proc. Visualization '94*, pages 40–45, 1994.
[32] M. Sharir and P.K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.

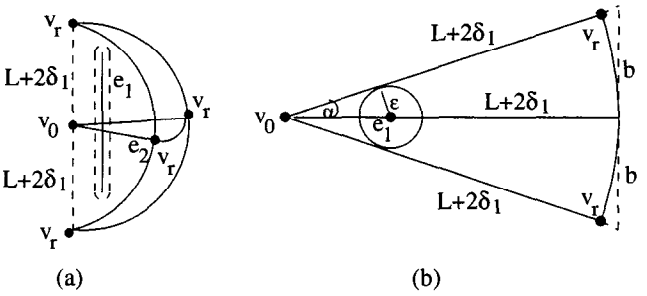

Figure 4: (a) Forbidden loci for v_r. (b) Cross section of (a).

[33] J.R. Shewchuk. Adaptive robust floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18:305–363, 1997.
[34] K. Sugihara. On finite-precision representations of geometric objects. *J. Comput. Syst. Sci.*, 39:236–247, 1989.
[35] K. Sugihara and M. Iri. Two design principles of geometric algorithms in finite-precision arithmetic. *Appl. Math. Lett.*, 2(2):203–206, 1989.
[36] J.D. Weld and M.C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *Internat. J. Robot. Res.*, 9(5):105–117, 1990.
[37] P.G. Xavier. Fast swept-volume distance for robust collision detection. In *Intl. Conf. Robotics and Auto.*, pages 1162–1169, 1997.
[38] C.K. Yap. Symbolic treatment of geometric degeneracies. *J. Symbolic Computation*, 10:349–370, 1990.
[39] C.K. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7:3–23, 1997.

A Appendix: Forbidden Loci and a Bound on δ

In this appendix we give more details on the shape and volume of forbidden loci which induce degeneracies. The computation of forbidden loci is tedious and involves many cases and sub-cases, and therefore we give only two examples here, which illustrate the techniques used while computing all the forbidden loci. The full details of forbidden loci for the local step are given in [28]. The full details for both local and global steps are given in [27]. The two examples here are both concerned with the degeneracy of type *edge-edge*, showing how different the computations are in the local step compared to the global step. Notice that although the analysis is complicated, it is being used only for proving that our theory is correct; in practice the perturbation scheme remains simple and easy to implement.

Recall that δ_1, δ_2 denote the maximum perturbation radius for the local and global steps respectively. Let $P \oplus Q$ denote the Minkowski sum of P and Q and let $P \ominus Q$ denote the Minkowski sum of P and $-Q$, where $-Q := \{-q \mid q \in Q\}$. Let K, L, V be as defined in Section 3.6. Let $\mathcal{L} := L + 2\delta_1$ be the maximum edge length after perturbing both endpoints.

We describe the loci F_3 mentioned in Sections 3.4 and 3.5. In order to distinguish between F_3 of the local and global steps, we call them here F_{local} and F_{global} respectively.

Forbidden Loci for Degeneracy of Type *Edge-edge* in the Local Step Let $P = \{P_1, P_2, \ldots, P_m\}$, s_i $(1 \le i \le m)$, Q_r $(1 \le r \le s_i)$ be as defined in Section 3.4. Recall that we remove the degeneracies in each polyhedral surface locally, by an incremental procedure where we add the vertices of

each surface one by one and if a degeneracy is detected we only perturb the last vertex that has been added. Suppose the procedure is at stage r. We would like to find a location for v_r so that no local degeneracy of type *edge-edge* is incurred; namely, for every edge e_1 in Q_{r-1} and for every edge e_2 created by v_r, we would like to have $d(e_1, e_2) > \varepsilon$. Let $v_0 \in Q_{r-1}$ denote the other endpoint of e_2. Notice that the locations of e_1 and v_0 are static at stage r, while the location of v_r has not been determined yet.

In order to create an edge $e_2 = (v_0, v_r)$, v_r has to be located inside the ball $B(v_0, \mathcal{L})$. We aim to identify the subvolume of this ball, in which locating v_r induces a degeneracy of type *edge-edge*. In Figure 4(a) we show the forbidden loci for v_r; depicted is a sphere slice of radius $\mathcal{L}$, centered at v_0. The 'axis' of the slice is parallel to e_1. Locating v_r inside the slice creates an edge e_2, which will be too close to e_1 if it intersects $e_1 \oplus B(0, \varepsilon)$. Hence the forbidden loci are inside a sphere slice where the planar facets are tangent to $e_1 \oplus B(0, \varepsilon)$. Let $Slice(e_1, v_0, v_r)$ denote the forbidden loci for v_r referring to edge e_1 and vertex v_0.

Hence $F_{local} = \{Slice(e, v, v_r) \mid e \in Q_{r-1}, v \in Q_{r-1}\}$, where e is an edge and v is a vertex incident to v_r, denotes all the forbidden loci for v_r.

Our goal is to compute the volume of F_{local} and we do that by first computing the volume of $Slice(e_1, v_0, v_r)$. We denote the angle of the slice by 2α. The way of obtaining α is by analyzing the cross section of the sphere slice (drawn inside the slice in Figure 4(a)), which looks like a sector. The sector itself is depicted in Figure 4(b).

Using the sector as it is induces a rather unwieldy formula. In order to simplify the calculations we have bounded the sector by a triangle, as depicted in Figure 4(b). Let the length of the bounding edge of the sector be $2b$. Our worst case volume is when α reaches its maximum possible value. Since $\sin\alpha = \frac{\varepsilon}{d(v_0, e_1)}$, it follows that α becomes larger as $d(v_0, e_1)$ becomes smaller. According to our robustness definition, $d(v_0, e_1) > \varepsilon$, but such a distance might give a sphere slice which is too big. Therefore we define a constant $\rho > \varepsilon$, and make sure that for each edge e and vertex v in Q_r we have $d(v, e) > \rho$. The procedure for ensuring such a bound is simple and described in [28]. (Later on we see that it is enough that $\frac{\varepsilon}{\rho}$ is very small, and therefore choose the value of ρ to be $\varepsilon^{0.5}$). We now compute the worst case value of α, and infer a worst case value of b:

$$\sin\alpha = \frac{b}{\sqrt{\mathcal{L}^2 + b^2}} \leq \frac{\varepsilon}{\rho} \implies b^2 \leq \frac{\varepsilon^2}{\rho^2 - \varepsilon^2}\mathcal{L}^2$$

Let Δ denote the bounding triangle of the sector.

$$Area(\Delta) = 2 \cdot \frac{b\mathcal{L}}{2} \leq \frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\mathcal{L}^2$$

Hence a bound for the area proportion between the sector and a circle of the same radius is:

$$\frac{2\alpha}{2\pi} = \frac{Area(sector)}{\pi\mathcal{L}^2} \leq \frac{Area(\Delta)}{\pi\mathcal{L}^2} \leq \frac{\varepsilon}{\pi\sqrt{\rho^2 - \varepsilon^2}}$$

Using the ratio $\frac{2\alpha}{2\pi}$, we can bound the sphere slice volume:

$$Volume(sphere\ slice) = \frac{2\alpha}{2\pi} \cdot \frac{4}{3}\pi\mathcal{L}^3 \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\mathcal{L}^3$$

The volume of F_{local} is the product of the following:

- The maximum number of edges incident to v_r, $3V - 3$.

  (It suffices to use the maximum degree of a vertex, but we use a crude bound here, which is the maximum number of edges in a surface. $3V - 3$ is obtained by Euler's formula applied to the case of possibly no outer facet, and assuming triangular facets).
- The maximum number of edges which a single edge might intersect, $2K$.

  (An edge can intersect the interior of at most two edges in a facet).
- The volume of a sphere slice.

Hence an upper bound on the volume of forbidden loci for v_r, for a degeneracy of type *edge-edge*, is:

$$Volume(F_{local}) \leq 8\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}KV(L + 2\delta_1)^3$$

Forbidden Loci for Degeneracy of Type *Edge-edge* in the Global Step Let $T = \{T_1, ..., T_l\}$, $con(T_j)$, M_j $(1 \leq j \leq l)$ be as defined in Section 3.5. Recall that we remove the degeneracies by an incremental procedure where we add the terrains one by one and if a degeneracy is detected we only perturb the last terrain that has been added. Suppose the procedure is at stage j. We aim to find a location for T_j so that no degeneracy of type *edge-edge* is incurred.

We describe here the forbidden loci induced by the new location of T_j relative to M_{j-1}. We do not detail forbidden loci induced by $con(T_j)$, since their computation is similar to the one described for the local step.

For every edge e_1 in M_{j-1} and for every edge e_2 in T_j, we would like to have $d(e_1, e_2) > \varepsilon$. The set $\{e \oplus B(0, \delta_2)\}$ where e is an edge in M_{j-1}, denotes the volume that any edge of T_j must not intersect. Hence the set $F_{global} = \{e_1 \oplus B(0, \delta_2)\} \ominus \{e_2 \mid e_2 \in T_j\}$ (where e_1 is an edge in M_{j-1}) denotes the forbidden translation vectors of T_j. The Minkowski sum of two edges is a parallelogram, with area bounded by $\mathcal{L}^2$. The Minkowski sum of a parallelogram and a ball $B(0, \delta_2)$ is an expanded parallelogram (see Section 4.1), with volume bounded by:

$$2\varepsilon\mathcal{L}^2 + 4 \cdot \frac{1}{2}\mathcal{L}\pi\varepsilon^2 + 3 \cdot \frac{4}{3}\pi\varepsilon^3$$

The volume consists of three components: a rectangular prism, four half cylinders, and three balls at the corners.

The volume of F_{global} is the product of the following: (i) The maximum number of edges in a terrain, $3V - 3$, (ii) The maximum number of edges which an edge might intersect, $2K$, (iii) The volume of an expanded parallelogram. Hence the forbidden volume is bounded as follows:

$$Volume(F_{global}) \leq 12KV\varepsilon((L + 2\delta_1)^2 + \pi(L + 2\delta_1)\varepsilon + 2\pi\varepsilon^2)$$

Computing a Bound on δ After computing all the forbidden loci, we obtain a list of volumes, where each volume is a function of $\varepsilon, \rho, K, L, V, \delta_1$ and δ_2. Assuming $K \geq 10$, $V \geq 10$, $L \geq 10$, $\delta_1 \leq L$, $\delta_2 \leq L$ and $\varepsilon \leq 10^{-4}$, and assigning $\rho = \varepsilon^{0.5}$, we convert the volumes to the form $c_1\pi\varepsilon^{0.5}KL^3V$ for the local step, and $c_2\pi K^3\varepsilon^{0.5}L^3V$ for the global step, where c_1, c_2 are constants. For example, the volume of F_{local} computed above is converted in the following way:

$$\begin{aligned} Volume(F_{local}) &\leq 8\frac{\pi}{3}(\sqrt{1.0001}\varepsilon^{0.5})K \cdot 27L^3 \cdot V \\ &\leq 72.1\pi\varepsilon^{0.5}KL^3V \end{aligned}$$

Solving the inequality $c_1\pi\varepsilon^{0.5}KL^3V < \frac{1}{2} \cdot \frac{4}{3}\pi\delta_1^3$, we get $\delta_1 > (\frac{3}{2}c_1)^{\frac{1}{3}}\varepsilon^{\frac{1}{6}}K^{\frac{1}{3}}LV^{\frac{1}{3}}$, and in a similar way we get $\delta_2 > (\frac{3}{2}c_2)^{\frac{1}{3}}\varepsilon^{\frac{1}{6}}KLV^{\frac{1}{3}}$.