

# Controlled Perturbation for Arrangements of Polyhedral Surfaces <sup>\*</sup>

Sigal Raab<sup>†</sup>

Dan Halperin<sup>‡</sup>

June 2002

## Abstract

We describe a perturbation scheme to overcome degeneracies and precision problems for algorithms that manipulate polyhedral surfaces using fixed precision 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 size of the perturbed surfaces (namely the overall number of vertices, edges, and faces) is linear in their original size. A trade-off exists between the magnitude of the perturbation and the efficiency of the computation—the larger the allowed perturbation, the smaller the running time. Our perturbation scheme can be used by any application that manipulates polyhedral surfaces and needs robust input, typically in areas like solid modeling, manufacturing and robotics. Our work is based on [15] which handles the case of spheres, extending the scheme to the more difficult case of polyhedral surfaces. A by product of our work, which we believe is of independent interest, is the analysis and categorization of all possible degeneracy types arising in arrangements of polyhedral surfaces.

## 1 Introduction

There are two major causes for robustness problems in the implementation of geometric algorithms: careless use of floating point arithmetic and degenerate input data. Floating point arithmetic problems are often caused by degenerate (or near-degenerate) data. Hence, both problems are closely related. These problems have been extensively studied in recent years; for surveys on robustness in geometric algorithms see, e.g., [17, Chapter 4],[25],[32].

---

<sup>\*</sup>This work is a part of Sigal Raab's M.Sc. thesis, prepared under the supervision of Prof. Dan Halperin from Tel-Aviv University and Prof. Amihod Amir from Bar-Ilan University. Work reported in this paper has been supported in part by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces), by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Center for Geometric Computing and its Applications), and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

<sup>†</sup>School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. Part of this work has been carried out while S.R. was a student at the Bar Ilan University. Email: raab@tau.ac.il.

<sup>‡</sup>School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. Email: danha@tau.ac.il.

Geometric algorithms are usually designed and proved 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, implementations of geometric algorithms often use finite precision floating point arithmetic, where the precision is determined by the hardware of the specific system and may vary 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 lie on 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. (For a formal definition of the term *degeneracy* see, e.g., [4].) 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 resources required by the algorithm, lead to difficult and boring case analysis, and produce cluttered code.

A seemingly straightforward solution to robustness problems is the use of exact arithmetic, see, e.g., [1],[2],[3],[6],[19],[34]. Although the recent decade has seen tremendous progress in exact arithmetic computation, it is still often slower compared to floating point arithmetic and requires special methods for non-rational values (roots of polynomials, trigonometric functions, etc.).

In many cases a computation will be correct even if floating point arithmetic has been used. This leads to floating point filters [11],[19],[29] where all computation steps that give correct results are done with 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 devised in order to avoid degenerate cases is *symbolic perturbation* [8],[9],[26],[33]. Perturbations are performed only symbolically by replacing each coordinate of every input geometric object by a polynomial in an indeterminate which is thought of as taking arbitrarily small values, while maintaining consistency of the input data. Symbolic perturbation methods require exact arithmetic.

In a different approach sometimes referred to as *heuristic epsilons* or *epsilon-tweaking* [25], the programmer chooses an arbitrary small value  $\varepsilon$ . Whenever the absolute difference between two values is less than  $\varepsilon$ , they are considered equal. Scientific justification is seldom given for choosing any specific  $\varepsilon$  value. However, 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 term *finite precision approximation* for a family of algorithms that perform slight perturbations of the input data in order to overcome robustness problems [10],[12],[13],[15], [16], [20],[30],[31]. Unlike symbolic perturbation, this is an actual (slight) modification of the input geometric objects. 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 an approximate manner. In both cases the computerized model is known not to be totally precise.

One approach of this family is *snap rounding* [12],[13],[16],[21], where the data of an arrangement of line segments are slightly changed, so that every vertex of the arrangement is positioned at a center of a pixel and no intersections occur other than in those centers. In [10] *snap rounding* is extended to three dimensions, with time and storage complexity of  $O(n^4)$ , where  $n$  is the total number of vertices, edges, and facets in a polyhedral subdivision.

A recent approach is described in [15], where degeneracies in a special collection of spheres (modeling the atoms of a molecule) are completely removed by a relatively simple scheme. The approach of [15] 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 [15] is the basis for our work, in which we completely remove all degenerate cases from a collection of polyhedral surfaces, and obtain output size of  $O(n)$  (with a small constant of proportionality), where  $n$  is the number of facets in the input surfaces.

The rest of the paper is organized as follows. In the next two sections we define the terminology that we will use in the paper and expose the key ideas of our approach. The technical details of our scheme are presented in Section 4, followed by complexity analysis in Section 5. Experimental results are reported in Section 6. In Section 7 we relax some of the simplifying assumptions introduced earlier. Concluding remarks and suggestions for further research are outlined in Section 8. We then describe additional technicalities in the appendices.

## 2 Preliminaries

In this section we review the basic terminology that we will use throughout the paper, define the objects that our algorithm handles, and explain what properties of the input objects we aim to preserve.

**Incidence Relation** We use the term *incidence relation* to describe the fact that two entities  $c_1$  and  $c_2$  such that  $c_1$  is on the boundary of  $c_2$ , are neighbours along the surface (we also say that they are *topologically connected*). For example, in Figure 1, vertex  $v_2$  is incident to edges  $e_1, e_2, e_5$  and to facets  $f_1, f_2$ ; edge  $e_1$  is incident to edges  $e_2, e_4, e_5$ , to vertices  $v_1, v_2$  and to facets  $f_1, f_2$ ; facet  $f_1$  is incident to vertices  $v_1, v_2, v_4$ , to edges  $e_1, e_2, e_3, e_4, e_5$  and to facet  $f_2$ . Although vertex  $v_5$  is geometrically touching edge  $e_2$ , there is no incidence relation between them because they are not defined as topologically connected.

**Polyhedral Surface Representation** Our perturbation scheme manipulates polyhedral surfaces that are manifolds and have only triangular facets. We allow self-intersections of the surfaces (allowing intersections does not contradict the manifold property since there are no incidence

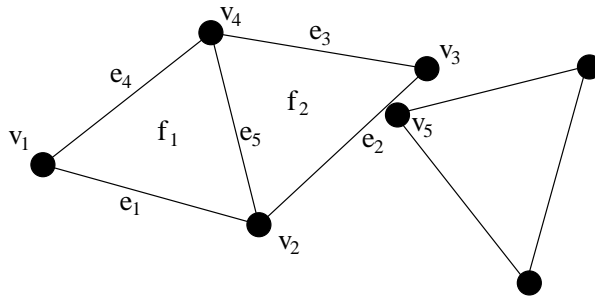


Figure 1: Illustration of incidence relations.

relations between the intersecting facets) as well as intersections among surfaces. A polyhedral surface may have an arbitrarily large number of facets.

Our representation of a polyhedral surface is similar to Kettner's [18] where the data structure describing a polyhedral surface maintains its geometric and topological information. The geometric information consists of all vertices, where each vertex is represented by three coordinates with constant bit length. The topological information consists of incidence relations between edges, vertices and facets. For example, each edge points to an incident facet, vertex and edge. The output of our algorithm is the set of perturbed surfaces, using the same data structure and using the same number of bits to represent each coordinate.

**Arrangements** Let  $P = \{P_1, P_2, \dots, P_m\}$  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 more details on arrangements of surfaces see [14],[27]. Notice that while most work on 3D arrangements of geometric objects assumes that each object has constant descriptive complexity (e.g., triangles), we deal with the more complicated form of arrangements of (arbitrary size) polyhedral surfaces.

**Taking the Spheres Scheme One Step Ahead** We base our algorithm on the paradigm presented by Halperin and Shelton [15]. While [15] deals with arrangements of spheres, here 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 the relative position of two or more spheres, for polyhedral surfaces we have to remove degeneracies internal to one surface, in addition to removing degeneracies induced by relative positions of two or more surfaces. Moreover, 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.

**Preservation of Topological Characteristics** Since our scheme modifies the input surfaces, it is most important to define what characteristics of the input polyhedral surfaces are preserved during the perturbation process. As we will see shortly, there is a (typically small) parameter  $\delta > 0$  so that no feature of any surface is moved during the perturbation process by more than

$\delta$  from its original placement. Another aspect that needs to be determined is how much of the original topological structure (structure, for short) of the input surfaces we preserve.

Three approaches for the preservation of structure come to mind: (i) Treat the input as a collection of triangles that have no relation to each other (as is done in [15] 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 within a surface, but do not keep intersection data as part of the 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 part of the structure and must be maintained during the perturbation process. The three approaches are illustrated in 2D in Figure 2, where for approach (i) the input is treated as a collection of segments with no incidence relations between them, for approach (ii) incidence relations for each input triangle are kept, but no intersection data is kept, and for approach (iii) even the intersection points are part of the structure.

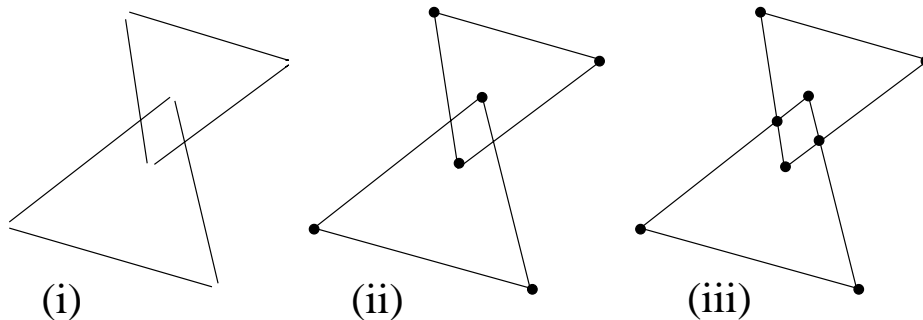


Figure 2: A 2D illustration of the three approaches to preservation of topological characteristics.

We further clarify the difference between approaches (ii) and (iii) by distinguishing between edges and intersection segments. In approach (ii) every edge is part of the structure and its incidence relations must be preserved, i.e., its incident facets are not disconnected from it even if undergoing perturbation. In contrast, an intersection segment is not a part of the structure to be preserved, and therefore a segment that might have originally existed in the arrangement induced by 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 part of the structure that needs to be preserved 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 topological incidence relations within each surface (see below). Approach (iii) preserves more topological 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 (e.g., motion planning) care about the incidence topology of each surface locally, more than about the topology of the full subdivision (as we discuss in more detail below).

**Motivation: Swept Volume** Our work has been primarily motivated by swept volume computation [24]. 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*. See Figure 3 for an example of a volume that has degeneracies, created by a triangle swept along a trajectory which intersects itself.

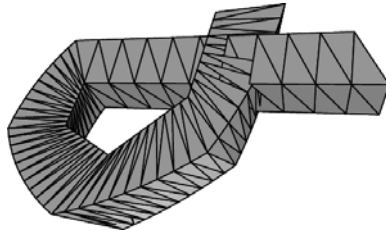


Figure 3: A swept volume of a triangle with degeneracies that arise when its trajectory intersects itself. Our perturbation scheme removes those degeneracies.

Among many geometric applications that could possibly benefit from our scheme, we choose to mention swept volume computation and (more generally) motion planning problems. These illustrate why we have chosen to design our algorithm according to approach (ii) of topological characteristics preservation. In these applications, incidence relations are crucial and therefore should be preserved—they bound for example swept regions or forbidden regions in the configuration space of a robot motion planning problem. On the other hand if users accept a  $\delta$  size deviation of the input (when we perturb some of the data), it only seems natural that some of the inter-surface relations will be modified. Indeed such changes may modify the overall structure of the arrangement of the surfaces. However, we believe that these changes are secondary in importance and in most cases would not matter (assuming again that the application at hand allows a deviation of the input by  $\delta$ ).

**Arrangements vs. Collections of Surfaces** Our algorithm is a preprocessing stage before computing the arrangement of the input surfaces. The actual construction of the arrangement is not part of our algorithm; it should be taken care of by another package—see for example [28]. Notice that although we keep referring to the underlying arrangement of the surfaces in order to guide the perturbation process, the output of the algorithm is a *collection* of (perturbed) surfaces and not the subdivision that they induce.

### 3 Key Ideas

We target geometric applications that manipulate polyhedral surfaces. These applications are better off with degeneracy-free input; for instance, a vertex should not touch a non-incident facet. Since we are using fixed-precision 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.

In order to define such a potential degeneracy formally, we use a *resolution parameter*,  $\varepsilon > 0$ .  $\varepsilon$  is a small positive real number. Two non-incident features of polyhedral surfaces are assumed too close (and therefore potentially degenerate) whenever they are not at least  $\varepsilon$ -away from each other (i.e., the distance between two polyhedral features is less than or equal to  $\varepsilon$ ). The full list of potential degeneracies for a collection of polyhedral surfaces is given in Section 4.2.

We assume  $\varepsilon$  to be given as an input parameter according to the machine precision and to the maximum length of an edge in a polyhedral surface. The value of  $\varepsilon$  depends also on the depth of the expression tree [25]. In our setting and underlying applications (construction of the arrangement, swept volume computation) this tree’s depth is a small constant, thus  $\varepsilon$  is a constant that can be determined and bounded independent of the complexity of the surfaces.

Ideally we would have liked to perturb the features of the arrangement by at most  $\varepsilon$  from their original placement. However, the features are interdependent and we cannot guarantee such a bound on the perturbation size. Therefore we define a perturbation radius  $\delta$ , which depends on  $\varepsilon$  and on other parameters of the input, and is proved to be small enough. A potentially degenerate polyhedral surface is perturbed by at most  $\delta$ , namely, each vertex of the surface is moved by Euclidean distance of at most  $\delta$ .

We call our scheme “controlled perturbation”. It is controlled in two aspects. First, by determining the size of  $\delta$  we control the running time of the perturbation scheme and set a trade-off between the magnitude of the perturbation and the efficiency of the computation. Second, unlike in  $\varepsilon$ -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 which permit such perturbation, since often their precision is limited to start with (due to measurement limitations or approximate modeling).

Our scheme makes some simplifying assumptions, most of which are relaxed later at Section 7. These assumptions preclude certain ‘pathological’ input instances and do not influence any of the data that we have examined.

## 4 Controlled Perturbation

In this section we expose the perturbation algorithm, describe the degeneracy types that we remove and detail the way of removing those degeneracies.

### 4.1 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 pairwise non-incident facets.

$d(\mathbf{p}_1, \mathbf{p}_2)$  denotes the Euclidean distance between  $p_1$  and  $p_2$ , where  $p_1, p_2$  are three-dimensional points.

$d(G_1, G_2)$  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  $d(G_1, G_2) = \min\{d(p_1, p_2) \mid p_1 \in G_1, p_2 \in G_2\}$ .

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

## 4.2 Inventory of Degeneracies

We next define the degeneracies to be removed from a collection of polyhedral surfaces. Throughout the paper, whenever describing a degeneracy, we use square brackets for terms that would have been used had we been using exact arithmetic.

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 Appendix A 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 degeneracies:

*vertex-facet* A vertex is too close to [touches] a non-incident facet.

*edge-edge* An edge is too close to [intersects] a non-incident edge.

*edge-segment* An edge is too close to [intersects] a non-incident segment.

*facet-intr3* A facet is too close to [contains] a non-incident intr3 [four facets intersect in a point].

All the degeneracy types can be either local or global (see Figure 4):

**Local degeneracies** 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 degeneracies** Degeneracies involving two or more polyhedral surfaces, e.g., a vertex in one surface is too close to [touches] a facet in another surface.

## 4.3 Algorithm Overview

Our perturbation process consists of the following steps:

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



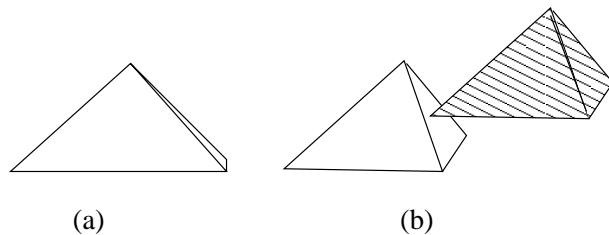


Figure 4: (a) Local *vertex-facet* degeneracy. (b) Global *vertex-facet* degeneracy.

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

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 units to perturb (e.g., a whole polyhedral surface) will achieve the simplicity goal, but will not remove internal degeneracies. In order to solve this conflict we choose to remove 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 degeneracies are removed.

## 4.4 Local Step

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

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  $P = \{P_1, P_2, \dots, P_m\}$  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  $v_1, v_2, \dots, v_{s_i}$  be an ordering of the vertices in  $P_i$ . Let  $Q_r$  denote the data structure describing the perturbed  $P_i$  after processing vertices  $v_1, \dots, v_r$ ,  $1 \leq r \leq s_i$ .  $Q_r$  contains the possibly perturbed vertices  $v_1, v_2, \dots, v_r$ , and all the edges and facets of the current polyhedral surface  $P_i$ , whose incident vertices are in  $\{v_1, v_2, \dots, v_r\}$ .

Let  $\delta_1$  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:

- $LI_1$  Any vertex in  $Q_r$  has been moved by distance at most  $\delta_1$  from its original placement.
- $LI_2$   $d(v, f) > \varepsilon$  for every pair of non-incident vertex  $v$  and facet  $f$  in  $Q_r$ .
- $LI_3$   $d(e_1, e_2) > \varepsilon$  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  $LI_2$  and  $LI_3$  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 4.6 we describe how we determine  $\delta_1$  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 invariants  $LI_1$ . Placing  $v'_r$  outside the forbidden loci  $F_2$  and  $F_3$  guarantees invariants  $LI_2$  and  $LI_3$ . Notice that if  $v_r$  already keeps the invariants, no perturbation will take place. The region  $F_3$  is, for example, a union of sphere slices and is described in Appendix B. Bounding the volume of the forbidden loci is tedious and involves many cases and sub-cases, and therefore we give only two examples in Appendix B. All forbidden loci are described in detail in [24]. Notice that bounding the volumes of forbidden loci is done only ‘on paper’ in order to determine the value of  $\delta_1$ . The actual perturbation algorithm is very simple once  $\delta_1$  has been determined.

## 4.5 Global Step

In this step we remove global degeneracies and local 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 degeneracies, thus each perturbation unit is now degeneracy free.

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

**stitching** Stitch formerly incident terrains by creating *connectors* between them. See Figure 5.

We next describe the three sub-steps in more detail.

**Partitioning into xy-Monotone Surfaces** A surface is  $xy$ -monotone (also called *terrain*) if every line orthogonal to the  $xy$ -plane intersects the surface in at most one point. It is guaranteed that no local 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 such that no facet is vertical. In our implementation we use a fairly straightforward partitioning algorithm which will not be discussed here.

Partitioning is always done on a sequence of incident edges. Let us call this sequence the *partitioning polyline*. Notice that edges and vertices in the partitioning polyline are duplicated, where one edge/vertex becomes a part of the first partitioned terrain, and its copy becomes a

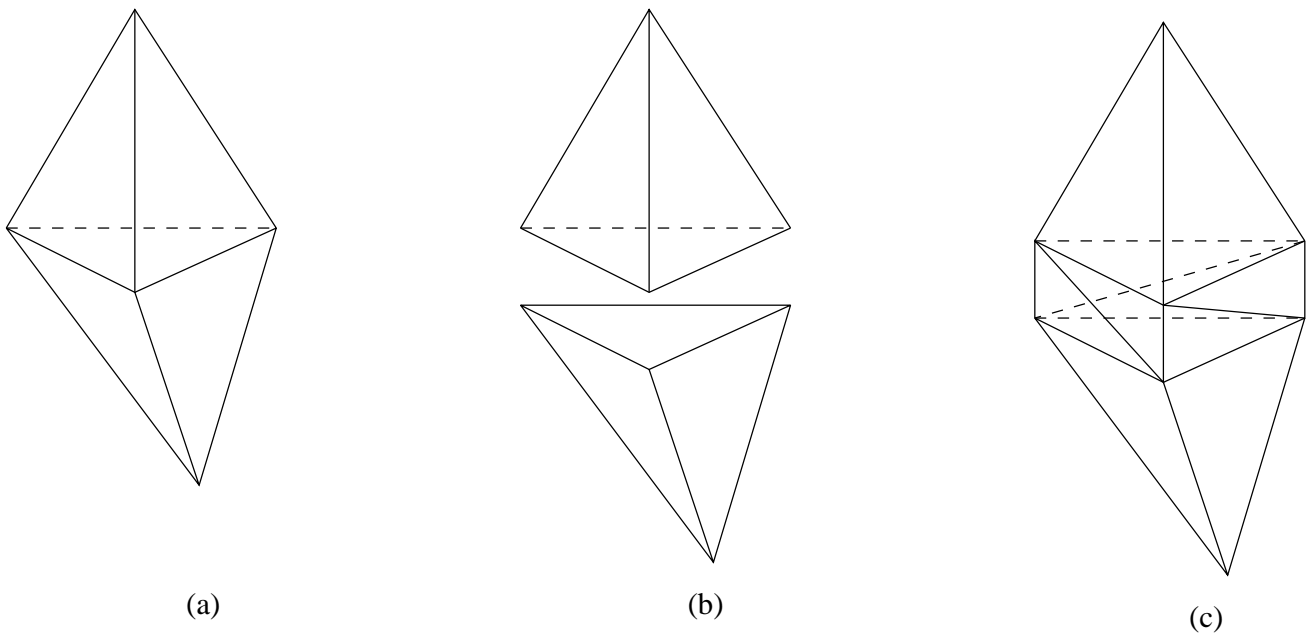


Figure 5: (a) A polyhedral surface. (b) Partitioning into  $xy$ -monotone surfaces and perturbing one of them. (c) Creating connectors (in reality the connectors are much thinner; here they are enlarged for clarity).

part of the second one. Later on, those terrains might be perturbed, thus the two copies may be different geometric entities.

**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 (as a single rigid entity). In addition to testing the validity of the (possibly) perturbed terrain, we make sure that no degeneracies are induced by the new *connectors* (see definition below) which stitch the terrain we are currently handling to incident terrains that were added previously.

Let  $T = \{T_1, \dots, T_l\}$  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 after processing terrains  $T_1, \dots, T_j$ ,  $1 \leq j \leq l$ .  $M_j$  is a collection of polyhedral surfaces comprised of the possibly perturbed  $T_1, \dots, T_j$  and the connectors  $con(T_2), \dots, con(T_j)$ .

Let  $\delta_2$  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  $T_1, \dots, T_j$  has been moved by at most  $\delta_2$ , and that there are no degeneracies in  $M_j$ . In detail, after the completion of stage  $j$ , the following invariants are guaranteed:

- $GI_1$  Any vertex of any terrain in  $M_j$  has been moved by distance at most  $\delta_2$  from its original placement.
- $GI_2$   $d(v, f) > \varepsilon$  for every pair of non-incident vertex  $v$  and facet  $f$  in  $M_j$ .
- $GI_3$   $d(e_1, e_2) > \varepsilon$  for every pair of distinct non-incident edges  $e_1, e_2$  in  $M_j$ .
- $GI_4$   $d(e, s) > \varepsilon$  for every pair of non-incident edge  $e$  and segment  $s$  in  $M_j$ .
- $GI_5$   $d(f, i) > \varepsilon$  for every pair of non-incident facet  $f$  and intr3  $i$  in  $M_j$ .

The invariants  $GI_2, \dots, GI_5$  define forbidden loci  $F_2, \dots, F_5$  respectively. Let  $E_j := B(0, \delta_2) \setminus (F_2 \cup \dots \cup F_5)$ . We keep choosing points inside  $B(0, \delta_2)$  uniformly at random, until the chosen translation vector is inside  $E_j$ . Section 4.6 supplies more details about  $\delta_2$  and explains why the choice of  $\delta_2$  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  $GI_1$ . Choosing the translation vector outside the forbidden loci  $F_2, \dots, F_5$  guarantees invariants  $GI_2, \dots, GI_5$ . Notice that if  $T_j$  already keeps the invariants, no perturbation will take place. The region  $F_3$ , for example, is a union of Minkowski sums<sup>1</sup> of a 2D parallelogram and a ball  $B(0, \varepsilon)$ , and is described in Appendix B. All the forbidden loci are described in detail in [24].

**Stitching** A *connector* is created in the following way, depicted in Figure 5(c): An edge is created between each pair of partitioned vertices (along the partitioning polyline as described in the partitioning sub-step), 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 formerly incident terrains. If two originally incident terrains are not perturbed, then no connector is created and their original incidence relations are restored (the terrains are ‘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 ( $\delta_2$  is typically very small), a connector is 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

---

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

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  $\delta_2$ . Analyzing those loci is complicated and explained in detail in [24], but the implementation remains simple.

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

## 4.6 Choosing $\delta$

In both the local and global steps, we would like 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. See Figure 6. Let  $E := B \setminus F$  denote the valid placements. We are interested in choosing a point inside  $E$ .

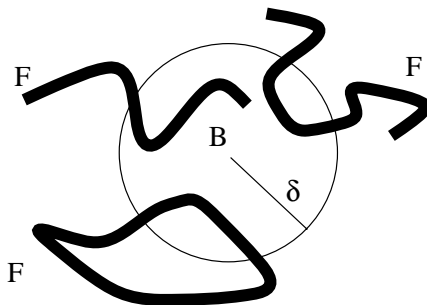


Figure 6: We would like to choose a point inside a ball, avoiding forbidden regions of the ball (2D illustration).

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 B) show that  $Volume(F)$  is always finite and that we need to choose  $\delta_1 = 6.31 \varepsilon^{1/6} K^{1/3} L V^{1/3}$  and  $\delta_2 = 2.15 \varepsilon^{1/6} K L V^{1/3}$ , 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  $\varepsilon \leq 10^{-4}$ .  $K$  is a small constant in all realistic cases. Notice that when  $K$  is bounded, both  $\delta_1$  and  $\delta_2$  are independent of the input size  $n$ .  $K$  is further discusses in Section 4.7 and Section 7.

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

## 4.7 Simplifying Assumptions

In this Section we list some simplifying assumptions, that differentiate the practical case from the theoretical case. All the assumptions made here pertain to practical cases, and make the exposition of our ideas simpler. However, since we wish our scheme to cover any conceivable input, including ‘pathological’ ones, we relax these assumptions in Section 7.

**Definitions of  $K$  and  $D$**  In a collection of polyhedral surfaces, we assume that the number of facets that a single facet intersects is bounded by a small constant  $K$ . This assumption is based on a large number of geometric models that we have examined, which intersect each other only in a controlled and limited manner.

In order to achieve better performance we discretize the three-dimensional space into grid cubes of edge length  $L + 2\delta_1 + 2\varepsilon$ . In Section 5.1 we define an expanded facet to be the Minkowski sum of the facet and  $B(0, \varepsilon)$ . We denote by  $D$  a bound on the number of expanded facets intersecting a grid 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 that we examined. The experimental results supply some findings about  $K$  and  $D$ .

Notice that the values of  $K$  and  $D$  are determined during the running time: While perturbing some of the entities, the number of intersections (and hence  $K$  and  $D$ ) might change. These values are checked throughout the entire perturbation process and  $K$  and  $D$  are determined accordingly. In abuse of notation, in our experiments we have examined the static values that have been obtained before the perturbation has been run. From our experience, those values are good estimates of the actual values of  $K$  and  $D$ , since  $\delta$  is very small and therefore the number of intersections hardly changes during running time.

It is possible to create ‘pathological’ input data, where  $K$  or  $D$  are much bigger than the static values obtained before the perturbation is run, or even where  $K$  or  $D$  are not constants and might be as large as  $n$ . It is important to emphasize that in such cases our perturbation scheme and the complexity analysis are still valid, with greater significance to the values of  $K$  and  $D$  in the bound on  $\delta$  and in some of the complexity results given below.

**The Number of Facets Affecting a Single Facet** Let  $\Psi$  be the maximum number of facets *affecting* a single facet, meaning that for a given facet  $f$  we look for the maximum number  $\psi(f)$  of facets that could have *any* effect on the perturbation distance of  $f$  at any stage of our algorithm, like intersecting  $f$ , being  $\varepsilon$  close to  $f$ , etc. We define  $\Psi$  to be the maximum  $\psi(f)$  over all facets  $f$  in  $P$ .

For the time being we assume that a certain facet can be affected only by the facets that intersect it, i.e., that the number of facets affecting a certain facet is bounded by  $K$ . This assumption turns out to be good in practice (namely  $K$  is a good estimate for  $\Psi$ ). In Section 7 we relax this assumption.

**Stitching Procedure** Notice that the stitching procedure works well as long as at most two terrains meet in one point. 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 and the perturbation size in this case; we leave the  $\delta$  bound proofs of this special case for future research, and assume in the analysis below that this case does not arise.

**The Bits Issue** As mentioned earlier the input vertices of the polyhedral surfaces as well as the output vertices of the perturbed surfaces are represented with fixed precision. Thus we can view the vertices of the surfaces as lying on the vertices of a fine grid whose unit size is determined by the coordinate precision. Our last assumption is that  $\varepsilon$  is larger than the unit  $u$  of the coordinate grid, say  $\varepsilon > 1000u$ . By this assumption we achieve the following: given a ball  $B$  in which we look for a valid perturbation and the forbidden region  $F$  inside it, the ratio  $Volume(F)/Volume(B)$  approximates very well the ratio  $\#(F)/\#(B)$ , where  $\#(A)$  is the number of coordinate grid vertices lying in the region  $A$ . Since all the forbidden regions in our analysis are equal to or larger than a ball of radius  $\varepsilon$ , this assumption guarantees that the difference between  $Volume(F)/Volume(B)$  and  $\#(F)/\#(B)$  is negligible and it is subsumed in the constants introduced in the analysis of  $\delta$ . This is the only assumption that is not relaxed in Section 7. We are currently studying the choice of  $\varepsilon$  looking for safe yet better (smaller) bounds.

## 5 Complexity Analysis

The analysis in this section is carried out under the simplifying assumptions presented in Section 4.7. Complexity bounds that change when the simplifying assumptions are relaxed are detailed in Section 7.

### 5.1 Time Complexity

Recall that  $L$  is the maximum length of an edge in the input data,  $\delta_1$  is the maximum perturbation distance in the local step (where the perturbation unit is one vertex) and  $\varepsilon > 0$  is the given resolution parameter. In order to achieve better performance we discretize the 3D space into grid cubes of edge length  $L + 2\delta_1 + 2\varepsilon$  each. We use the term *expanded entity* for the Minkowski sum of an entity and the ball  $B(0, \varepsilon)$ . See Figure 7(a). We choose the length of a grid cube edge according to the maximum length of an expanded edge that has possibly been perturbed: We add  $2\delta_1$  to  $L$  because each endpoint of the edge has possibly been perturbed by at most  $\delta_1$ , and then we add  $2\varepsilon$  because the edge is expanded. An expanded entity can intersect at most 8 grid cubes (i.e., a cube whose edge length is double the size of a grid cube edge, see Figure 7(b)).

For each grid cube we maintain a list of the expanded entities intersecting that cube. As already mentioned above we assume that the number of expanded facets intersecting a single cube is bounded by a constant  $D$ . See Figure 7(c). 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 segments, `intr2` and `intr3` in one grid cube is bounded by  $DK$ ,  $2DK$ , and  $D\binom{K}{2}$  respectively.

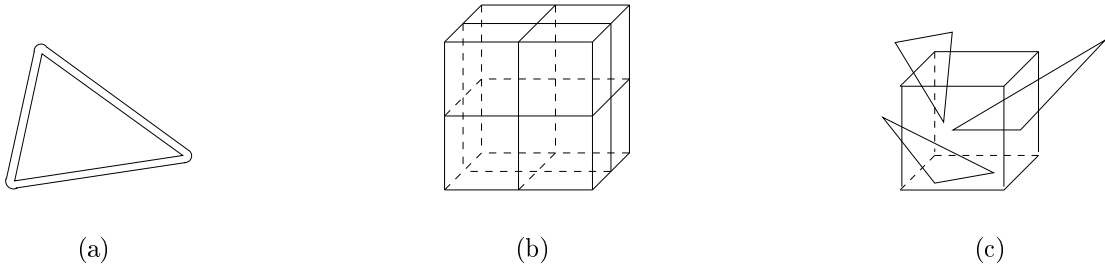


Figure 7: (a) An expanded facet. (b) An expanded entity can intersect at most 8 grid cubes. (c) We assume that the number of expanded facets intersecting a single grid cube is bounded by a constant  $D$ .

### 5.1.1 Local Step

Together with the choice of  $\delta_1$  (as described in Section 4.6), the guaranteed expected time of this step is  $O(nD)$ , as we explain next.

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.

For removing degeneracies of type *edge-edge*, we look at all the cubes that an expanded edge intersects (at most 8) and check for intersection with the edges in those cubes (at most  $3D$  per cube). Since there are at most  $3n$  edges, we get that at most  $72nD$  combinations of two edges need to be checked.

In addition, the special cases *vertex-vertex* and *vertex-edge* are checked within each facet locally. There are 3 combinations of two vertices in a facet, and 3 combinations of a vertex and a non-incident edge in a facet. Therefore we get that at most  $6n$  combinations need to be checked for each of those degeneracies.

### 5.1.2 Global Step

Together with the choice of  $\delta_2$  (as described in Section 4.6), the guaranteed expected time of this step is  $O(n \log^3 n + nDK^2)$ , as we explain next.

**Partitioning Sub-Step** The time complexity of the partitioning sub-step is  $O(n \log^3 n)$ :

The partitioning sub-step is done by an incremental procedure, where breadth first search is applied over the facets that 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  $T_i$ . The facet fits the terrain if the following are true (where  $\bar{o}$  denotes the projection of an object  $o$  onto the  $xy$ -plane): (i)  $\bar{v}$  is not inside the incident facet in  $\bar{T}_i$  (tested in  $O(1)$  time). (ii) The whole terrain  $\bar{T}_i$  is not inside  $\bar{f}$  (tested in  $O(1)$  time). (iii)  $\bar{e}_1, \bar{e}_2$  do not intersect any of



the edges of  $\bar{T}_i$  (tested in  $O(\log^3 n)$  time using a dynamic data structure for ray shooting [5]).

**Perturbation Sub-Step** For removing degeneracies of type *vertex-facet* and *edge-edge*, there are at most  $24nD$  and  $72nD$  combinations (respectively) that need to be checked, as explained for the local step.

For removing degeneracies of type *edge-segment*, we look at all the cubes that an expanded edge intersects (at most 8) and check for intersection with the segments in those cubes (at most  $DK$  per cube). Since there are at most  $3n$  edges, we get that at most  $24nDK$  combinations of an edge and a segment need to be checked.

For removing degeneracies of type *facet-intr3*, we look at all the cubes that an expanded facet intersects (at most 8) and check for intersection with the intr3 in those cubes (at most  $D\binom{K}{2}$  per cube). Since there are at most  $n$  facets, we get that at most  $8nD\binom{K}{2}$  combinations of a facet and an intr3 need to be checked.

In addition we have to detect all the segments, intr2 and intr3 in the data structure:

For detecting segment entities, we look at all the cubes that an expanded facet intersects (at most 8) and check for intersection with the other facets in those cubes (at most  $D$  per cube). Since there are at most  $n$  facets, we get that at most  $8nD$  combinations of two facets need to be checked.

For detecting intr2 entities, we look at all the cubes that an expanded facet intersects (at most 8) and check for intersection with the edges in those cubes (at most  $3D$  per cube). Since there are at most  $n$  facets, we get that at most  $24nD$  combinations of a facet and an edge need to be checked.

For detecting intr3 entities, we look at all the cubes that an expanded facet intersects (at most 8) and check for intersection with the segments in those cubes (at most  $DK$  per cube). Since there are at most  $n$  facets, we get that at most  $8nDK$  combinations of a facet and a segment need to be checked.

In summary, the asymptotic complexity of the perturbation sub-step is dominated by the removal of the *facet-intr3* degeneracy, which requires  $O(nDK^2)$  time.

## 5.2 Storage Complexity

### 5.2.1 Output Size

The only increase in the space of the input data is due to creation of 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  $D$ .

### 5.2.2 Working Storage

There are two types of working storage that are needed by our scheme. The first one is during the partitioning sub-step, where we need  $O(n \log n)$  space for the ray shooting procedure [5]. The second one is during the perturbation sub-step where we need to maintain lists of intersection entities: segment and intr2 entities require  $O(nK)$  space, and intr3 entities require  $O(nK^2)$  space. Altogether we obtain a working storage of  $O(n \log n + nK^2)$ .

We summarize the performance of our perturbation scheme in the following theorem.

**Theorem 5.1** *Given a collection  $P$  of polyhedral surfaces with a total number of  $n$  triangular facets, and a resolution parameter  $\varepsilon > 0$ , fulfilling the assumptions of Section 4.7, 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  $P$ , and  $D$  is the maximum number of expanded facets intersecting a grid cube. A perturbation is considered valid if at the end of it every geometric entity is at least  $\varepsilon$ -away from any other geometric entity and topological incidence relations are preserved, and it is obtained by moving each vertex by Euclidean distance at most  $\delta$  from its original placement.  $\delta$  is a parameter that depends on  $\varepsilon$ ,  $K$ , the maximum edge length in  $P$ , and the maximum number of vertices in one polyhedral surface.*

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.

## 6 Experimental Results

We have implemented our scheme and in this section we present our experimental results, which show that our scheme is effective in practice.

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  $l$  respectively. 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 report 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 [22]. Many of the software classes used in our implementation belong to the CGAL<sup>2</sup> library.

In our experiments we do not rely on the theoretical bounds on  $\delta$  derived earlier but rather fix  $\delta$  values that are much smaller than the theoretical ones. The success of the perturbation scheme with these small bounds on examples with many degeneracies shows that the theoretical bounds are crude and in practice much smaller values are sufficient.

---

<sup>2</sup>CGAL: Computational Geometry Algorithms Library, see <http://www.CGAL.org>

In Table 1 we give time results for several swept volumes (swept volumes are defined in Section 2), which differ in the number of input facets. In the results we specify the time needed to remove degeneracies. In order to supply a basis for comparison, we use  $\varepsilon=1e-8$  and  $\delta=1e-4$  for all the experiments, and choose all examples to be the swept volume of the same generator moving along the same trajectory. We use a swept volume application that 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 effect on  $\bar{d}$  is striking. The swept volume of 302 facets is depicted in Figure 3.

$n$	$K$	$\bar{k}$	$\bar{d}$	$L$	$\bar{l}$	$V$	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 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.

In Table 2 we show the tradeoff between the magnitude of the perturbation and the efficiency of the computation, introduced by varying values of  $\delta$  and  $\varepsilon$ . The perturbation procedure is run on a swept volume of 842 facets, with  $\varepsilon$  that varies in the range  $[1e-3, 1e-8]$  and  $\delta$  that varies in the range  $[1e-1, 1e-6]$ . This table illustrates that, as stated above, the practical  $\delta$  values are much smaller than the values obtained in the theoretical analysis.

$\delta$	$\varepsilon$	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 degeneracies, introducing the tradeoff between the magnitude of the perturbation and the efficiency of the computation. The examined object is of size 842 where  $\bar{k}$  is 0.4 and  $\bar{d}$  is 31.3.

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 location limited to a cube of a given size, and a random orientation. 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).

	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 degeneracies, and  $K$  values, for a collection of 500 random triangles. Triangles edge lengths are specified in the left column. Cube edge lengths (bounding triangles) are specified in the upper row.

In Table 4 we compare the time ratio for removal of 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 total running time. (Notice that *con* in the table is a part of the global step). The first volume has no degeneracies at all, the second one (depicted in Figure 8) 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. A one dimensional illustration of the fourth volume is depicted in Figure 9. We use  $\epsilon=1e-8$  and  $\delta=1e-3$  for all the experiments, and choose all of the examined swept volume sizes to be between 916 to 932 facets. Although the sizes of the four models are almost the same, their  $\bar{d}$  values vary significantly, as their densities are different.

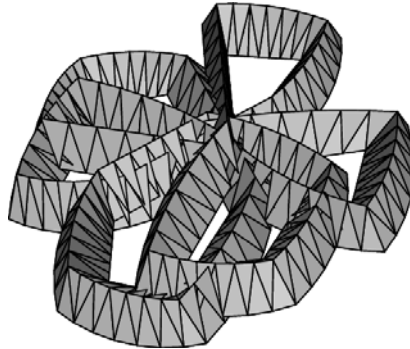


Figure 8: A swept volume inducing many degeneracies. The dense intersection area in the center makes it difficult to find a valid perturbation.



Figure 9: A one dimensional illustration of a swept volume, where the  $xy$ -partitioning process results in many terrains and therefore many connectors.

file name	$\bar{k}$	$\bar{d}$	<i>total</i>	$\frac{\textit{local}}{\textit{total}}$	$\frac{\textit{global}}{\textit{total}}$	$\frac{\textit{con}}{\textit{total}}$
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 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.

## Implementation Issues

- In the complexity analysis of the partitioning sub-step we used a dynamic data structure for ray shooting. In our implementation however, in the partitioning sub-step, it is substituted by a fairly straightforward algorithm which is much simpler and performs well in practice. We do not discuss the details of this algorithm here.
- As already mentioned in Section 4.7,  $K$  and  $D$  are dynamic values which are determined during the running time. In abuse of notation, in our experiments we have examined the static values that have been obtained before the perturbation has been run. From our experience, those values are good estimates of the actual values of  $K$  and  $D$ , since  $\delta$  is very small and therefore the number of intersections hardly changes during running time.

## 7 The General Case

In this Section we present the general case, relaxing the assumptions made in Section 4.7 and improving the perturbation scheme that has been presented so far. We do not make the general case an integral part of the perturbation scheme, since it is needed only for ‘pathological’ inputs. We present the general case in order to show that we have a solution even for the most difficult (synthetic) inputs.

**The number of facets affecting a single facet** Recall that facet  $f_1$  is *affected* by facet  $f_2$  if  $f_2$  affects the perturbation distance of  $f_1$ . The obvious ways for affecting a facet are by intersecting it or being  $\varepsilon$  close to it, but there are also indirect ways, as described in the example below.

In our scheme we assume that a certain facet can be affected only by the facets that intersect it, i.e., we assume that the number of facets affecting a certain facet is bounded by  $K$  (recall that  $K$  bounds the number of facets intersecting a single facet).

Here is an example in which the maximum number of facets affecting a single facet is significantly greater than  $K$ . Suppose that every planar layer in Figure 10 is a collection of  $N \times N$  (say

$N = 100$ ) planar equilateral triangles that are  $1.5\varepsilon$  away from each other. Recall that  $L \gg \varepsilon$ . Suppose further that the input data consists of  $N$  such planar collections, ordered vertically one on top of the other, with vertical space of  $1.5\varepsilon$  between every two layers. So far we described a perfectly valid collection of  $N^3$  triangles. Once this collection is given as input to our scheme, there will not be a single change in it. Now suppose the  $N^3 + 1$ st triangle of the input data is a horizontal triangle that is located in the center of the collection of  $N^3$  triangles that have already been processed. This last triangle induces a degeneracy of type *vertex-facet*, and therefore needs to be perturbed. Its  $K$  value is at most 26, but the minimum perturbation location that assures a valid placement is outside the collection of  $N^3$  triangles. In other words, the number of triangles affecting the last triangle is not bounded by  $K$  or even close to  $K$ .

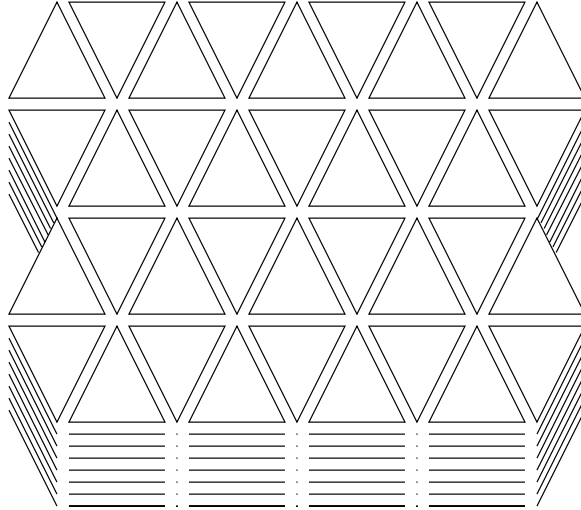


Figure 10: A collection of triangles where the maximum number of facets affecting a single facet is greater than  $K$ .

We now turn into improving our scheme such that it will be valid for any number of facets affecting a single facet.

Let  $\Psi$  be the maximum number of facets affecting a single facet. Namely, for a given facet  $f$  we look for the maximum number  $\psi(f)$  of facets that could have *any* effect on the perturbation distance of  $f$  at any stage of our algorithm.  $\Psi$  denotes the maximum  $\psi(f)$  over all facets  $f$  in the collection  $P$  of input polyhedral surfaces.

$\Psi$  replaces  $K$  in the formula of  $\delta$ , so we get that  $\delta_1 = 6.31 \varepsilon^{1/6} \Psi^{1/3} L V^{1/3}$  and  $\delta_2 = 2.15 \varepsilon^{1/6} \Psi L V^{1/3}$ . Recall that  $\delta = \delta_1 + \delta_2$ .

$\Psi$  is close to  $K$  in most reasonable inputs but may be as large as  $n$  in ‘pathological’ inputs. However,  $\Psi$  is *not* known in advance. Therefore we perform a binary search for  $\Psi$ . We start by guessing  $\Psi_0 := 1$ , and if a valid location is not found for every degenerate input data (i.e., a constant number, say four, of guesses result in an invalid location), then we multiply the guess by two, i.e.,  $\Psi_i := 2\Psi_{i-1}$ . The algorithm is guaranteed to stop once our guess is greater than or equal to the actual  $\Psi$ .

The binary search increases the running time by a factor of  $\log \Psi$ , and the new expected

running time is  $O(n \log^3 n \log \Psi + nDK^2 \log \Psi)$ . Notice that the working storage and the output size do not change.

**Bounds Assuming Worst Case for  $K$ ,  $D$  and  $\Psi$**  In some ‘pathological’ cases,  $K$ ,  $D$  and  $\Psi$  might be as large as  $n$ . In these cases, the running time can be as large as  $O(n^4 \log n)$ , and the working storage can be as large as  $O(n^3)$ . Notice that the output size remains  $O(n)$ .

The bound on  $\delta$  is also affected if  $K$  or  $\Psi$  are large, namely  $\delta_1 = 6.31 \varepsilon^{1/6} n^{1/3} LV^{1/3}$  and  $\delta_2 = 2.15 \varepsilon^{1/6} n LV^{1/3}$ .

## 8 Conclusion

We have presented a perturbation scheme for a collection of polyhedral surfaces in 3D space. The scheme removes all degeneracies and thus overcomes precision problems. It is useful for applications that use such polyhedral surfaces, allowing them the usage of finite precision arithmetic and sparing them the need of handling degenerate cases. Our scheme takes the scheme proposed for spheres [15] one step ahead, to the more difficult case of polyhedral surfaces. We have solved the difficulty of removing both local and global degeneracies by introducing the usage of terrains, and have taken into consideration that polyhedral surfaces may have arbitrarily many degrees of freedom.

We have an additional phase in our perturbation scheme, which is omitted here. It is fully described in [23] and [24]. In the additional phase we remove degeneracies which are induced by the vertical decomposition algorithm [7], 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.

**Discussion: Our Scheme vs. Three Dimensional Snap Rounding** 3D snap rounding of polyhedral subdivisions [10] 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 to compare, since we follow approach (ii) described in Section 2, while [10] follows approach (iii). However, since 3D snap rounding is the most similar work that we are aware of that solves this sort of problem in finite precision approximation, it is still worth a discussion.

Snap rounding handles subdivisions (i.e., approach (iii)) and therefore handles a topological structure which is more complicated than the one that 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 [10]. 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 implemented the scheme in a software package and proved its validity in practice). The output size of the algorithm in [10] is  $O(n^4)$ , where our scheme has  $O(n)$  output size. Last but not least, our perturbation scheme totally removes all

kinds of degeneracies, while snap rounding removes most degeneracies but allows one type of degeneracy, namely the incidence of many segments at a vertex. 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. Furthermore, in snap rounding a vertex can be extremely close to a non-incident edge [21].

**Future Research** We propose several directions for future research:

1. Extend the perturbation scheme to arbitrary (non-manifold) subdivisions.
2. Optimize the partitioning sub-step, in the aspect of minimizing the number of terrain border edges (and therefore minimizing the length of connectors). The optimization problem seems hard, and we are also interested in an approximation to the optimization.
3. Prove that  $\delta$  is bounded also for the case mentioned in Section 4.7, where more than two terrains meet at a point. We remind the reader that we have been able to handle such cases in practice, but we do not have a theoretical guarantee here.

## Acknowledgements

The authors wish to thank Iddo Hanniel, Sarel 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 [18] which have been used in the experiments.

## References

- [1] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparata, 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.
- [2] 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.
- [3] H. Brönnimann and M. Yvinec. Efficient exact evaluation of signs of determinants. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 166–173, 1997.
- [4] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 16–23, 1994.
- [5] 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.



- [6] K. L. Clarkson. Safe and effective determinant evaluation. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 387–395, 1992.
- [7] M. de Berg, L.J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. *Discrete Comput. Geom.*, 15:35–61, 1996.
- [8] H. Edelsbrunner and E. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. on Graph.*, 9(1):66–104, 1990.
- [9] I.Z. Emiris, J. Canny, and R. Seidel. Efficient perturbations for handling geometric degeneracies. *Algorithmica*, 19(1-2):219–242, September 1997.
- [10] S. Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 116–125, 1998.
- [11] 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.
- [12] 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.
- [13] D. Greene and F. Yao. Finite resolution computational geometry. In *Proc. 27th IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.
- [14] 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.
- [15] D. Halperin and C.R. Shelton. A perturbation scheme for spherical arrangements with application to molecular modeling. *Comput. Geom. Theory Appl.*, 10:273–287, 1998.
- [16] J. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.*, 13(4):199–214, 1999.
- [17] C. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, San Mateo, California, 1989.
- [18] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 146–154, 1998.
- [19] K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
- [20] V.J. Milenkovic. Verifiable implementation of geometric algorithms using finite precision arithmetic. *Artificial Intelligence*, 37:377–401, 1988.
- [21] E. Packer and D. Halperin. Snap rounding revisited. In *Abstracts of the 17th European Workshop on Computational Geometry*, pages 82–85, Berlin, 2001. To appear in *Computational Geometry: Theory and Applications*.

- [22] S. Raab. Excerpts from M.Sc thesis, in [http://www.math.tau.ac.il/~raab/thesis\\_cites.html](http://www.math.tau.ac.il/~raab/thesis_cites.html). 1998.
- [23] S. Raab. Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 163–172, 1999.
- [24] S. Raab. Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes, in [http://www.math.tau.ac.il/~raab/master\\_thesis.ps](http://www.math.tau.ac.il/~raab/master_thesis.ps). M.Sc. thesis, Dept. Comput. Sci., Bar Ilan University, Ramat Gan, Israel, 1999.
- [25] S. Schirra. Robustness and precision issues in geometric computation. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 14, pages 597–632. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.
- [26] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1998.
- [27] M. Sharir and P.K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.
- [28] Hayim Shaul and Dan Halperin. Improved construction of vertical decompositions of three-dimensional arrangements. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 2002. To appear.
- [29] J.R. Shewchuk. Adaptive robust floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18:305–363, 1997.
- [30] K. Sugihara. On finite-precision representations of geometric objects. *J. Comput. Syst. Sci.*, 39:236–247, 1989.
- [31] K. Sugihara and M. Iri. Two design principles of geometric algorithms in finite-precision arithmetic. *Appl. Math. Lett.*, 2(2):203–206, 1989.
- [32] C. K. Yap. Robust geometric computation. In Jacob E. Goodman and Joseph O’Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 35, pages 653–668. CRC Press LLC, Boca Raton, FL, 1997.
- [33] C.K. Yap. Symbolic treatment of geometric degeneracies. *J. Symbolic Computation*, 10:349–370, 1990.
- [34] C.K. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7:3–23, 1997.

## A Inventory of Degeneracies

In this appendix we give the full list of degeneracies arising in arrangements induced by a collection of polyhedral surfaces, and explain why most of the degeneracies are special cases of four archetypes.

## A.1 Full List of Degeneracies

Recall that we 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. Recall that we use square brackets for terms that would have been used had we been using exact arithmetic.

*vertex-vertex* A vertex is too close to [overlaps] another vertex.

*vertex-edge* A vertex is too close to [touches] a non-incident edge.

*vertex-facet* A vertex is too close to [touches] a non-incident facet.

*vertex-segment* A vertex is too close to [touches] a non-incident segment.

*vertex-intr3* A vertex is too close to [overlaps] a non-incident intr3.

*edge-edge* An edge is too close to [intersects] a non-incident edge.

*edge-facet* An edge is too close to lying in a non-incident facet. [An edge and a facet are contained in the same plane and intersect each other].

*edge-segment* An edge is too close to [intersects] a non-incident segment.

*edge-intr3* An edge is too close to [contains] a non-incident intr3.

*facet-facet* A facet is too close to overlapping a non-incident facet. [Two facets are contained in the same plane and intersect each other].

*facet-segment* A segment is too close to lying in a non-incident facet. [A segment and a facet are contained in the same plane and intersect each other]. Also: [Three facets intersect in a line].

*facet-intr3* A facet is too close to [contains] a non-incident intr3. [Four facets intersect in a point].

*segment-segment* A segment is too close to [intersects] a non-incident segment.

*segment-intr3* A segment is too close to [contains] a non-incident intr3.

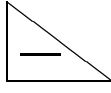
*intr3-intr3* An intr3 is too close to [overlaps] a non-incident intr3.

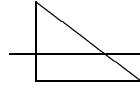
## A.2 Unifying Degeneracies

Some of the above pairs are special cases of other pairs:

*vertex-vertex*, *vertex-edge*, *vertex-segment*, *vertex-intr3* are all special cases of degeneracy of type *vertex-facet*.

*edge-facet* is divided into the following cases:

 is a special case of degeneracy of type *vertex-facet*.

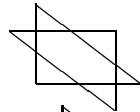
 is a special case of degeneracy of type *edge-edge*.

 is a special case of both degeneracy of type *vertex-facet* and *edge-edge*.

*edge-intr3* is a special case of degeneracy of type *edge-segment*.

*facet-facet* is divided into the following cases:

 is a special case of degeneracy of type *vertex-facet*.

 is a special case of degeneracy of type *edge-edge*.

 is a special case of both degeneracy of type *vertex-facet* and *edge-edge*.

*facet-segment* is divided into the following cases:

 is a special case of degeneracy of type *edge-segment*.

(Let  $s$  and  $f$  be a segment and a facet that induce degeneracy of type *facet-segment*. Let  $f_1$  and  $f_2$  be the facets that created  $s$ . A segment endpoint is always contained in an edge. Assume that the endpoint of  $s$  touches edge  $e$  of facet  $f_1$ .  $f$  and  $f_2$  intersect, and create another segment,  $s_1$ . Now if  $f$  and  $s$  induce degeneracy of type *facet-segment*, then  $e$  and  $s_1$  induce degeneracy of type *edge-segment*.)

 is a special case of degeneracy of type *edge-segment*.

 is a special case of degeneracy of type *edge-segment*.

*segment-segment* is a special case of degeneracy of type *facet-intr3*.

*segment-intr3* is a special case of degeneracy of type *facet-intr3*.

*intr3-intr3* is a special case of degeneracy of type *facet-intr3*.

After omitting the special cases, we end up with four types of degeneracies: *vertex-facet*, *edge-edge*, *edge-segment* and *facet-intr3*.

## B Forbidden Loci and a Bound on $\delta$

In this appendix we give more details on the shape and volume of forbidden loci that induce degeneracies. A bound on  $\delta$  is computed according to those volumes. 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.

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 for both the local and global steps are given in [24]. 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.

Recall that  $\delta_1, \delta_2$  denote the maximum perturbation distance 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 4.6. 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 4.4 and 4.5. In order to distinguish between  $F_3$  of the local and global steps, we call them here  $F_{local}$  and  $F_{global}$  respectively.

### B.1 Forbidden Loci for Degeneracy of Type *Edge-edge* in the Local Step

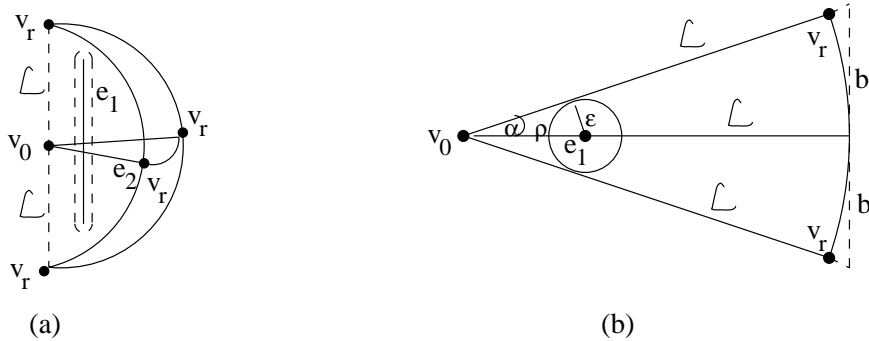


Figure 11: (a) Forbidden loci for  $v_r$ . (b) Cross section of (a).

Let  $P = \{P_1, P_2, \dots, P_m\}$ ,  $s_i$  ( $1 \leq i \leq m$ ),  $Q_r$  ( $1 \leq r \leq s_i$ ) be as defined in Section 4.4. 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 has reached 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 already fixed 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 sub-volume of this ball, in which locating  $v_r$  induces a degeneracy of type *edge-edge*, involving  $e_1$  and  $e_2$ . In Figure 11(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  $2\alpha$ . The way of obtaining  $\alpha$  is by analyzing the cross section of the sphere slice (drawn inside the slice in Figure 11(a)), which looks like a sector. The sector itself is depicted in Figure 11(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 11(b). Let the length of the bounding edge of the sector be  $2b$ . Our worst case volume is when  $\alpha$  reaches its maximum possible value. Since  $\sin \alpha = \frac{\varepsilon}{d(v_0, e_1)}$ , it follows that  $\alpha$  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 [24]. (Later on we see that it is sufficient that  $\frac{\varepsilon}{\rho}$  is very small, and therefore choose the value of  $\rho$  to be  $\varepsilon^{0.5}$ ). We now compute the worst case value of  $\alpha$ , 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  $\Delta$  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 be close to intersecting,  $2K$ .  
(An edge can be close to intersecting 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:

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

## B.2 Forbidden Loci for Degeneracy of Type *Edge-edge* in the Global Step

Let  $T = \{T_1, \dots, T_l\}$ ,  $con(T_j)$ ,  $M_j$  ( $1 \leq j \leq l$ ) be as defined in Section 4.5. 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 has reached 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 describe only the loci concerned with the degeneracy of type *edge-edge*. 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  $F_{global} = \{e_1 \oplus B(0, \delta_2) \mid e_1 \in M_{j-1}\} \ominus \{e_2 \mid e_2 \in T_j\}$  (where  $e_1, e_2$  are edges) 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 5.1), 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 four ball portions at the corners, whose union is bounded by the volume of three full balls.

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 be close to intersecting,

$2K$ , (iii) The volume of an expanded parallelogram. Hence the volume of  $F_{global}$  is bounded as follows:

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

### B.3 Computing a Bound on $\delta$

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  $\delta_2$ . 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  $\delta_2 > (\frac{3}{2}c_2)^{\frac{1}{3}}\varepsilon^{\frac{1}{6}}KL^{\frac{1}{3}}V^{\frac{1}{3}}$ .

Remark about the choice of  $\rho$ : As explained in Section 3,  $\varepsilon$  is an input parameter that depends (among others) on  $L$ . In particular,  $\varepsilon$  and  $L^2$  can be represented with the same scaling, i.e., with the same floating point exponent. Choosing  $\rho = \varepsilon^{0.5}$  (for  $\varepsilon \leq 10^{-4}$ ) means that  $\varepsilon < \rho \ll L$  and therefore  $\rho$  can also be represented with the same scaling as  $\varepsilon$  and  $L^2$ .