BAR-ILAN UNIVERSITY

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

Sigal Raab

supervised by:
Dan Halperin - Tel-Aviv University
Amihood Amir - Bar-Ilan University

Submitted in partial fulfillment of the requirements for the Master's degree in the Department of Mathematics and Computer Science, Bar-Ilan University

Ramat-Gan, Israel 1999

I deeply thank Prof. Dan Halperin for supervising the research and contributing useful ideas.

I thank Prof. Amihood Amir for supervising the research on behalf of the Bar-Ilan University.

I also wish to thank Iddo Hanniel, Gadi Fibich, Sariel Har-Peled, Oren Nechushtan, Lutz Kettner and Hayim Shaul for helpful discussions concerning the problems studied in this thesis. Lutz Kettner has also written the Polyhedron software classes [33] which have been used during the experiments.

Contents

Chapter 1

Introduction

In this chapter we give a survey about the problem which we try to solve, namely the problem of robustness in geometric algorithms in general, and in algorithms that manipulate polyhedral surfaces in particular. At the end of this chapter we describe the outline of the entire thesis.

1.1 Robustness in Geometric Algorithms

There are two major causes for robustness problems in geometric algorithms: the use of floating point arithmetic and degenerate input data. In this section we describe the problems that each of those subjects induces. Floating point arithmetic problems are often caused by degenerate input data; hence, both problems are closely related. For surveys on robustness in geometric algorithms see, e.g., [19],[25],[31, chapter 4],[40],[44].

1.1.1 Floating Point Arithmetic

Algorithms in the computational geometry literature 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 for a world with no numerical errors. In reality, most implementations of geometric algorithms are using finite precision floating point arithmetic. The reason for using floating point arithmetic is its speed and availability. The precision of floating point arithmetic is determined by the hardware of the specific system and it varies from one system to the other.

A floating point number consists of an *exponent* and a *mantissa*. Both are fixed length integers, which implies that they constitute a discrete set of representable rational numbers. Hoffman [31] enumerates three error types that are caused by 'squeezing' the infinite set of real numbers into the finite set of floating point numbers:

Conversion errors Converting the binary representations of the computer into decimal numbers and vice versa.

Roundoff errors Omitting digits that exceed the fixed length of a floating point number.

Digit-cancellation errors The difference of two nearly equal numbers a and b has fewer significant digits than does either a or b.

This 'squeezing' operation is not trivial and in many implementations it causes catastrophic errors in practice. The situation worsens when rounding errors accumulate. As a result we get programs that crash, loop forever, or simply compute wrong results.

The discussion above is true for any kind of algorithm, geometric or other. So why is it more complicated with geometric algorithms? The answer is that geometric algorithms are unique in that they operate on a mixture of numerical and combinatorial data. If we take a 3D polyhedral surface as an example, then we have numerical data, which is the vertices coordinates or the facets equations, but we also have combinatorial data, which is the incidence relations between facets, edges and vertices. In a floating point environment we might find out that the consistency between the geometric and combinatorial data was lost, and this is often the reason for many robustness problems.

Whenever we apply a geometric algorithm to a polyhedral surface, even a small rounding error might lead to major modeling problems: Areas of facets can come out negative, two facets that have a common edge (and thus only 'touch' each other) can become intersecting, and so on.

The numerical computations of a geometric algorithm are basically of two types: *predicates* and *constructions*. Predicates are associated with branching decisions and determine the flow of the algorithm, whereas constructions are needed to produce the output data. Thus approximations in the execution of constructions usually give acceptable results, as long as their maximum absolute error does not exceed the resolution required by the application. On the other hand, approximations in the evaluation of predicates may produce an incorrect branching of the algorithm, and lead to catastrophic consequences.

1.1.2 Degeneracies

The definition of a degenerate case varies from one algorithm to another. In general we may say that a degenerate case is detected whenever our algorithm needs to supply a special treatment. For example, if our algorithms defines a line according to two points, then three points which are collinear are a degenerate case. Not supplying a special treatment for such a case leads to finding three lines, defined by different pairs of points, overlapping.

Burnikel et al. [11] view a geometric algorithm as a decision tree where the decision nodes test the sign (+, −, or 0) of some function (usually a low degree polynomial) of the input variables. They define an instance x as *degenerate with respect to some algorithm A* if the computation of A on input x contains a test with outcome zero.

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 suffices that all of them are very close to the same line. In a floating point system, we may get wrong answers to predicates. For example, if three 2D points are close to being collinear, then the question "is point c on the right side of the line induced by a and b" might lead to several different answers, depending on the way of computation.

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 may cause an increase in the algorithm resources, lead to difficult and boring case analysis, and produce cluttered code.

1.2 Related Research

We next review some of the approaches that have been suggested by researchers to cope with robustness problems.

1.2.1 Exact Arithmetic

Since floating point arithmetic causes considerable problems, a seemingly straightforward solution is to use exact arithmetic [4],[9],[10],[14],[55]. Exact arithmetic is the most general technique that can guarantee the numerical reliability of an implementation of a geometric algorithm.

For exact arithmetic one may use rational numbers, represented by a numerator and a denominator. As a result of accumulated arithmetic operations, the values of the numerator and the denominator might become very big and lead to an integer overflow. This can be solved by using classes of unlimited length integers. Several software packages that supply such type of integers are listed in [44].

Exact arithmetic has two major disadvantages:

1. It is often slower compared to floating point arithmetic. Unlike floating point arithmetic, one cannot assume constant time for each arithmetic operation. The cost of an arithmetic operation depends on the context in which it is carried out.

2. Many geometric algorithms require irrational values. Calculating the Euclidean distance between two points requires a square root operation, rotating geometric entities requires trigonometric functions like sine and cosine, and so on. There are some exact algorithms that stick to rational numbers and approximate only the non-rational values, but then we cannot call them exact anymore.

1.2.2 Floating Point Filters

In many cases a program does not crash and gives correct results even if floating point arithmetic is used. This leads to floating point filters [21],[48], where we keep track of the accumulated errors, all computation steps that give correct result are done by floating point arithmetic, and only those steps which 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.

The decision whether an arithmetic calculation may be numerically incorrect is done by running some sort of test on the expression being evaluated concerning its input data. Passing the test means that the expression can be computed in floating point arithmetic. Failing the test means that a more exact reevaluation is needed.

Shewchuk [48] calls this paradigm *adaptive precision arithmetic.*

Schirra [44] describes two kinds of filters:

Static filters Compute error bounds a priori and need specific information on the input data to be available. For example, whether all input data are integers from a bounded range. This kind of filters require only little additional effort at run time.

Dynamic filters Compute an error bound on the fly simultaneously with the evaluation of expressions in floating point arithmetic. Their error estimation is much tighter than of a static filter, but more effort is required (in run time) on computing this estimation.

Thus a static filter makes arithmetic operations more efficient while a dynamic filter lets more floating point computations pass a test.

We mentioned earlier that the numerical computations of a geometric algorithm are basically of two types: *predicates* and *constructions*. Floating point filters are used only for the determination of predicates, for the purpose of ensuring that right decisions are taken during the flow of a program.

Notice the difference between heuristic epsilons (described shortly in Section 1.2.4) and floating point filters. In case of doubt, the former assumes a value of zero, while the latter invokes a more expensive computation, which finally leads to a correct decision.

1.2.3 Symbolic Perturbation

Symbolic perturbation was invented in order to remove degenerate cases [17],[18],[54]. Taking care of degenerate cases complicates algorithms, produces cluttered code, and leads to difficult and boring case analysis. Eliminating degeneracies would enable the use of algorithms that assume general position data, produce clear code, and ease the programming task.

The idea is that perturbations are performed only symbolically by replacing each coordinate of every input geometric object by a polynomial in ε, while maintaining consistency of the input data. The polynomials are chosen in such a way that the perturbed set goes towards the original set as ε goes to zero. The exact value of ε is not important and it is sufficient to assume that ε is positive and sufficiently small. The sign of the expression is given by the sign of the first nonzero polynomial coefficient, with coefficients taken in order of increasing powers of ε.

Edelsbrunner and Mücke call this paradigm *SoS* (Simulation of Simplicity) [17].

Fortune [19] mentions three disadvantages of this method:

1. It requires exact arithmetic.
2. A perturbation scheme exists only for a small number of problems.
3. Perturbing the input might sometimes lead to inaccurate results.

Burnikel et al. [11] assert that there is no reason for the panic induced by degenerate cases. They prefer dealing with each degenerate case separately rather than using the SoS paradigm. Their reasons are:

1. The perturbation algorithm induces overhead in running time.

2. The complexity of retrieving the perturbed answer to its original value is significant. They illustrate this claim by an example: Assume a problem of intersecting line segments. We may have the endpoint of some segment lying in the relative interior of some other segment. Perturbation may remove the point of intersection. This intersection is hard to retrieve from the perturbed output.

3. For some geometric algorithms, handling degeneracies directly is only a little more complex than algorithms assuming non-degenerate inputs. Moreover, in certain cases dealing directly with degeneracies can lead to improved running time.

So far we described approaches that require exact geometric computation, keep the input data unchanged and get the correct decisions out of predicates. Next we describe approaches that use geometric computation with imprecision, which means slightly changing the input data or treating them as if they were changed. Justification for such approaches are embedded in their description.

1.2.4 Heuristic Epsilons

In many implementations the method of dealing with the robustness problem is based on the rule of thumb, phrased by Schirra [44] as

If something is close to zero it is zero.

In this heuristics, the programmer chooses an arbitrary small value ε. Often whenever the absolute difference between two values is less than ε, they are considered equal.

Schirra suggests thinking of an arithmetic expression as of a labeled binary tree. Each inner node is labeled with a binary or unary operation. It has pointers to trees defining its operands. The pointers are ordered corresponding to the order of the operands. The leaves are labeled with constants or variables which are place-holders for numerical input values. Such a representation is called an *expression tree.* The *depth of an expression tree* is the length of the longest root-to-leaf path in the tree.

ε is chosen irrespective of the size of operands in a concrete expression or of the depth of an expression tree, and hence does not eliminate all kind of computation errors. Yap [55] calls this approach *epsilon tweaking.*

Scientific justification is seldom given for choosing any specific ε value, but for the pragmatic user, this approach works in many cases. Since it is very easy to implement, the user is willing to absorb the small amount of failures.

A major flaw in using heuristic epsilons is that the relation of equality is not transitive anymore. Guibas [25] gives the following illustrative example:

Consider the problem of sorting n numbers. Suppose that our comparison routine uses heuristic epsilons, i.e., it can report $a \geq b$, even though $a < b$, as long as $| a - b | < \varepsilon$. Now suppose we need to sort an array $A[1 \ldots n]$ and we use an algorithm that compares every consecutive pair of numbers. The array might be in reverse order ($A[1] > A[2] > \ldots > A[n]$), but any consecutive pair is very close ($| A[i] - A[i+1] | < \varepsilon$). In such a case, the algorithm reports $A[1] \leq A[2] \leq \ldots \leq A[n]$, thus concluding that the array is reported as sorted, although it is sorted backwards.

Guibas explains that a convex hull algorithm can get exactly into the same kind of difficulty depicted in Figure 1.1. An algorithm can process the points in (true) x-order and verify that each consecutive triplet is clockwise convex via an imprecise sign primitive, yet the resulting cycle can be arbitrarily far from being convex.

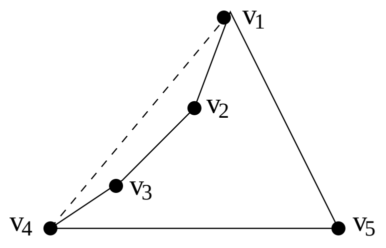


Figure 1.1: A wrong convex hull, illustrating that 'approximately clockwise convex' is not a transitive order

1.2.5 Finite Precision Approximation

This class of algorithms uses slight perturbations of the input data. Unlike symbolic perturbation, these are actual perturbations and not symbolic ones.

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 [29]), and hence possibly imprecise.

So far perturbations of the input were successful only for a small number of geometric algorithms. The success was achieved after usage of perturbations that were specifically designed for a given problem. A general theory showing how to implement geometric algorithms with perturbations is still a distant goal.

Milenkovic [38] suggests two perturbation methods:

Data normalization The key idea is to alter the structure and parameters of the geometric objects slightly so as to obtain an object for which all numerical tests are provably accurate. After this normalization process there are no arbitrary choices because calculation always gives a definitive and correct answer. In this method new degenerate cases are created, but their identification is definite. However, the algorithm that uses the data still has to take care of them. One technique being used is called *vertex shifting,* in which a vertex that lies within ε of another vertex is shifted to coincide with the other vertex. Another technique being used is *edge cracking*, where an edge e is replaced by a polygonal approximation of e. The edge e is cracked whenever there is a vertex that lies within ε of the edge, unifying the edge segments endpoints with those vertices. The data normalization method is probably the ancestor of *snap rounding* which is detailed below.

Hidden variable method The key idea is to choose a topological structure so that there exist infinite precision parameters for the object, close to the given finite precision parameter values, such that with the infinite precision values, the problem has the chosen structure. The approach is called hidden variable method since the topology of the infinite precision version is known but not its numerical values.

Sugihara[49] and Sugihara and Iri [50] use the name *finite precision arithmetic* or *finite precision representation* to describe algorithms that slightly perturb the input data. In [50] they describe two perturbation approaches:

Construction of an error-free world Construct a closed world in which topological structures of geometric objects are always determined precisely even in finite precision computation. The core of this approach is based on representing every 3D point as the intersection of three planes. Each plane is represented as the equation $a_ix + b_iy + c_iz + d_i = 0$, where all coefficients are integers satisfying $-L \leq a_i, b_i, c_i \leq L$ and $-L^2 \leq d_i \leq L^2$. Four ways for computing such coefficients which are close enough to the infinite data are detailed in [49]. In addition, basic operations are restricted to the form $A \leftarrow A * M(P)$, where A denotes a solid to be generated, P is a primitive solid, M a motion, and $*$ a set-theoretic operation such as union and intersection.

Giving the highest priority to topological consistency Avoid topological inconsistency by placing higher priority on logical consequence than on numerical judgment. A data with no degenerate cases is obtained as a result. In this approach they first define a set of rules that describe the topology of the data structure. Then the data structure is constructively built, where in each stage the numerical results might be slightly changed in order to keep the defined set of topology rules. The computation of each next step is done by using floating point arithmetic, and the result which is closest to the true desired data structure is chosen. They implement this approach in a new algorithm for constructing Voronoi diagrams for points given in the plane. The new program is robust against numerical errors, its average time complexity is linear in the number of input points (sites), and the structure of it is much simpler than the conventional, because it need not take care of degenerate cases.

Green and Yao [24], Goodrich et al. [23] and Hobby [30] study a paradigm called *snap rounding.* The input data for this paradigm is an *arrangement of line segments*, which is defined in [26],[28] as follows: Let $\mathcal{L}= \{\ell_1, \ell_2, ..., \ell_n)$ be a given collection of n line segments in the plane. We denote by $\mathcal{A}(\mathcal{L})$ the *arrangement* of $\mathcal{L}$, i.e., the decomposition of the plane into vertices, edges, and facets, induced by the line segments in $\mathcal{L}$. A vertex of $\mathcal{A}(\mathcal{L})$ is an intersection point of two line segments or an endpoint of a segment, an edge is a maximum connected relatively open portion of a line segment that does not meet any vertex, and a facet is a maximum connected open region of the plane not meeting any edge or vertex.

In the *snap rounding* paradigm, the data of an arrangement of line segments is changed as follows. A pixel containing a segment endpoint or an intersection point is called a *hot pixel*. All segments intersecting a hot pixel are re-routed to pass through its center. Intersection points are defined as new vertices, thus in the output no segment intersects the interior of another segment. However, each original input segment now possibly becomes a polyline (a polygonal curve). Since no line intersects the interior of another line, then most intersection types (like three lines intersecting in one point) are eliminated. One type of degeneracies is encouraged though, namely coincidence of vertices. In many cases segment endpoints which were not overlapping, are snapped to the center of the same pixel and thus are unified. Since vertex overlapping can happen only in a center of a pixel, the identification of such a degeneracy is definite, as in the *data normalization* method. See Figure 1.2.

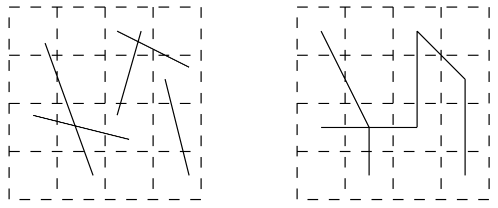


Figure 1.2: Snap rounding, before (left) and after

Goodrich et al. suggest an efficient plane sweep that performs the rounding and prevents additional crossings from being generated.

Fortune [20] extends the snap rounding paradigm to three dimensions. There is no straightforward extension for the snap rounding algorithm to 3D. Many difficulties have to be dealt with, and Fortune takes them into consideration. Fortune handles the input P, which is a polyhedral subdivision in $\mathbb{R}^3$ with a total of n cells. He shows that there is an embedding σ of the vertices, edges, and facets of P into a subdivision Q, where every vertex coordinate of Q is an integral multiple of $2^{-\lceil \log_2 n+2 \rceil}$. For each facet f of P, the Hausdorff distance in the L_∞ metric between f and $\sigma(f)$ is at most $3/2$. The embedding σ preserves or collapses vertical order on facets of P. The subdivision Q has $O(n^4)$ vertices in the worst-case, and can be computed in $O(n^4)$ time, where n, as mentioned above, is the total number of vertices, edges and facets in a polyhedral subdivision. In the algorithm, all edges of P are orthogonally projected onto the xy-plane. A planar arrangement is formed, snap rounded, and a triangulation T is created. The image of an edge is a polygonal chain within the triangulation, and the image of a facet is a sub-triangulation of T. The rounding of each facet f is obtained by lifting the image of f in T to three dimensions in such a way as to approximate f. By considering each cylinder over a vertex, edge, or triangle of T separately, Fortune ensures that the lifting preserves (or collapses) the vertical order on facets of P.

Halperin and Shelton [29] propose a perturbation scheme based on finite precision arithmetic. Their scheme completely eliminates degenerate cases in an arrangement of spheres, thus eases the programming task, and makes the code more clear and more efficient. The scheme is relatively simple and balances between the efficiency of computation and the magnitude of the perturbation. The potential degenerate cases are recognized, and each sphere is perturbed randomly a distance of at most δ from its original placement, to avoid degeneracies. The size of δ is determined according to the given resolution ε and a few more parameters. Halperin and Shelton prove that in the special type of arrangements they deal with (related to molecular models), δ is never too large. The algorithm runs in $O(n)$ time, where n is the number of spheres in the arrangement.

The paradigm presented by Halperin and Shelton is the basis for our research.

1.3 Thesis Outline

This thesis describes 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 $O(n \log^3 n + \text{nDK}^2)$ 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 [29] which handles the case of spheres, extending the scheme to the more difficult case of polyhedral surfaces perturbation. A paper describing this thesis [42] has been recently presented in the 1999 ACM Symposium on Computational Geometry[1].

The rest of the paper is organized as follows. In the next chapter we describe the main ideas and results of our work. In Chapter 3 we expose the controlled perturbation algorithm and describe technical details. We discuss complexity and performance in Chapter 4. In Chapter 5 we define swept volumes and describe the way in which they motivated the development of our scheme. We present experimental results in Chapter 6, and suggest an improvement in Chapter 7. A summary and suggestions for future research on the topic are given in Chapter 8. In Appendix A we complete details about the type of degeneracies which are removed by our scheme, and in Appendices B and C we complete technical details concerning shapes and volumes that arise in the perturbation scheme.

[1]A copy of the paper can be found in http://www.math.tau.ac.il/~raab/SoCG99.ps

Chapter 2

Preliminaries and Key Ideas

In this chapter we describe the main ideas and results of this thesis. We start with the main theorem, which is proved, slowly and thoroughly, throughout the entire thesis. Then we describe the key ideas, which are the core of the thesis, and after that we give some preliminaries about polyhedral surfaces, and describe the way in which we approach polyhedral surfaces.

2.1 The Main Theorem

Theorem 2.1 *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, and D is closely related to the maximum number of facets intersecting a grid cube. A perturbation is considered valid if at the end of it every geometric entity is at least ε-away from any other geometric entity and topologic incidence relations are preserved, and it is obtained by moving each vertex by Euclidean distance of at most δ from its original placement. δ is a parameter that depends on ε, K, the maximum edge length in P, 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 degeneracies which are unique to the swept volume application are removed.*

2.2 Key Ideas

Motivation The motivation for our scheme is the swept volume application. A *swept volume* is defined as the geometric space occupied by an object moving along a trajectory in a given time interval. Our swept volume application computes the boundary of a collection of three-dimensional polyhedra and employs *vertical decomposition* [15] 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 2.1 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.

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)

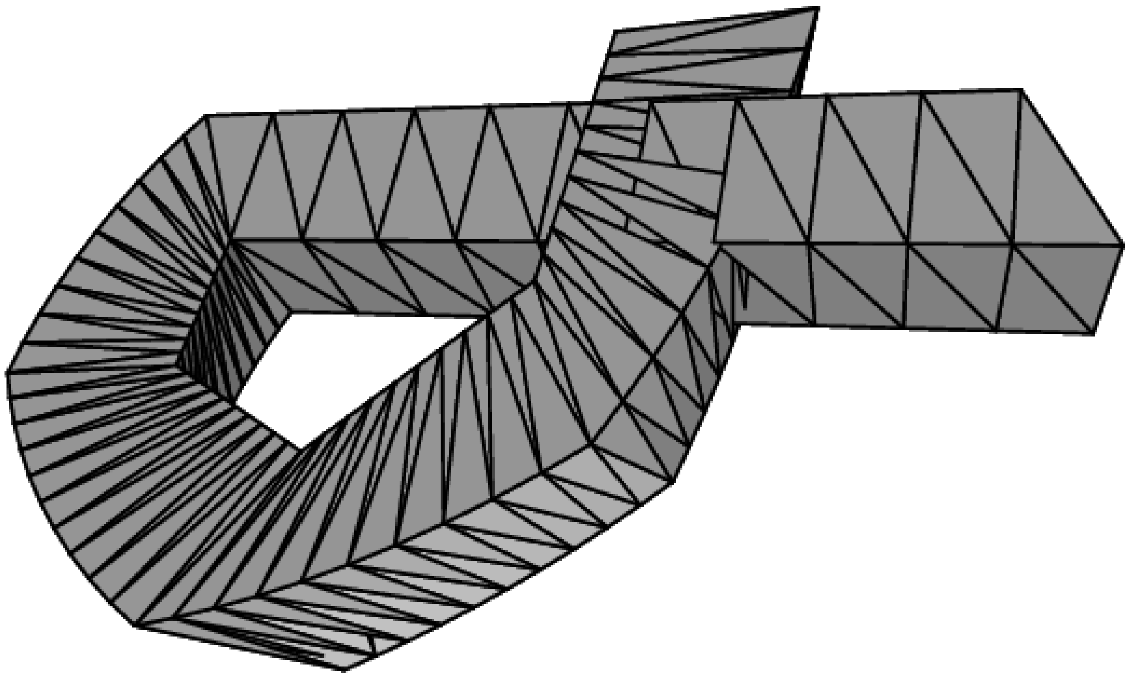


Figure 2.1: A swept volume of a triangle with degeneracies that arise when its trajectory intersects itself. Our perturbation scheme removes those degeneracies.

whenever they are not ε-away from each other (i.e., the distance between two polyhedral features is less than ε). 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 and to the maximum length of an edge in a polyhedral surface. The value of ε depends also on the depth of the expression tree [44]. 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 ε-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).

Our scheme makes some simplifying assumptions which are relaxed later at Chapter 7. Those assumptions concern certain 'pathological' input data and do not influence any of the real world data that has been examined by us.

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 Chapter 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 2.3). The output size $O(n)$ does not conceal any large constants, and does not depend on K or D.

2.3 Polyhedral Surfaces

Characteristics and Representation 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 [33] 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.

Arrangements Let $P = \{P_1, P_2, \ldots, P_m\}$ be a collection of m (possibly intersecting) polyhedral surfaces in $I\!R^3$. 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 [26],[46]. Notice that while most works about 3D arrangements of geometric objects assume that each object has constant descriptive complexity (e.g., triangles as in [15]), we deal with the more complicated form of arrangements of polyhedral surfaces.

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 fairly small constant K. This assumption is based on a large number of geometric models that we have 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 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 (related to 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 enough, since δ 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, 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.

Improvement Let Ψ be the maximum number of facets *influencing* 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 ε close to f, etc. We define Ψ to be the maximum $\psi(f)$ over all facets f in P.

At the first stage we assume that a certain facet can be influenced only by the facets that intersect it, i.e., that the number of facets influencing a certain facet is bounded by K. This assumption makes the exposition of our ideas simpler and turns out to be good in practice (namely K is a good estimate for Ψ). The main theorem is correct only under this estimation. Later on, in Chapter 7, we relax this assumption and address the actual quantity Ψ, which indeed in certain ‘pathological’ cases can differ greatly from K.

Taking the Spheres Scheme One Step Ahead We base our algorithm on the paradigm presented by Halperin and Shelton [29]. 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.

Preservation of Topologic Characteristics It is most important to define what characteristics of the polyhedral surface 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 [29] 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 are not 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 [20] 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 [20] follows approach (*iii*). However, since 3D snap rounding is the only previous work that 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 [20]. 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 [20] 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 Chapter 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.

Chapter 3

Controlled Perturbation

In this chapter we expose the perturbation algorithm, describe the degeneracy types which we remove and supply the technical details of removing those degeneracies.

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

$\boldsymbol{B(p, \mu)}$ 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 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 inherent 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 types of inherent degeneracies can be either local or global (see Figure 3.1):

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.

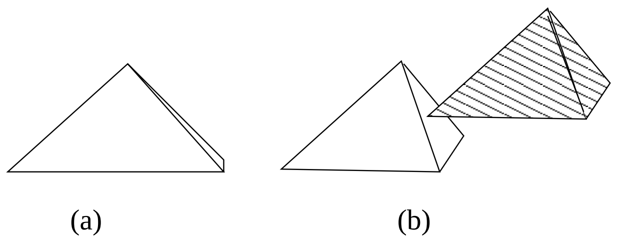


Figure 3.1: (a) local *vertex-facet* degeneracy. (b) global *vertex-facet* degeneracy.

Decomposition-induced degeneracies The list of degeneracies is defined mainly by the plane sweep stage in the vertical decomposition algorithm.

vertical-facet A facet is almost [exactly] vertical.

equal-x Two vertices have x values that are too close [equal].

concurrent-vertex-edge The xy-projection of a vertex and an edge are too close [intersect].

concurrent-intr2-edge The xy-projection of an intr2 and an edge are too close [intersect].

concurrent-three-edges The xy-projection of three edges intersect in three points which 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 is 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 $v_1, \ldots, v_r$, $1 \leq r \leq s_i$. 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, for example, a union of sphere slices. A detailed explanation of the shape of the objects contributing to F_2 and F_3 is given in Appendix B, at Section B.2.

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 3.2.

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

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 (Section 3.7). 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

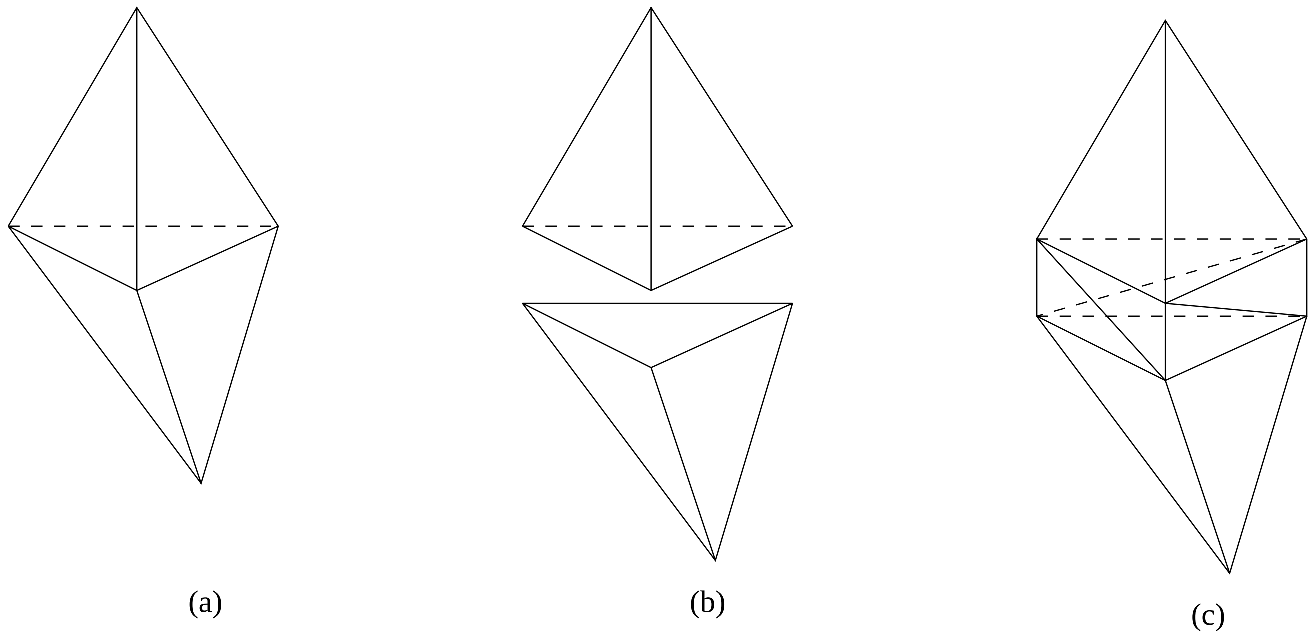


Figure 3.2: (a) Original polyhedral surface. (b) Partitioning into xy-monotone terrains and perturbing one of them. (c) Creating connectors (in reality the connectors are much thinner; here they are widened for clarity).

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 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, for example, a union of Minkowski sums[1] of a 2D parallelogram and a ball $B(0, \varepsilon)$. A detailed explanation of the shape of the objects contributing to $F_2, \ldots, F_5$ is given in Appendix B, at Section B.3.

Stitching A *connector* is created in the following way, depicted in Figure 3.2(c): An edge is created between each pair of partitioned vertices, inducing a parallelogram for every partitioned edge. The parallelogram is split

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

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 is created and their original adjacency 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 (δ_2 is 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 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 Appendix B, 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 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 minimum 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 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 3.3. Let $E := B \setminus F$ denote the valid placements. We are interested in choosing a point inside E.

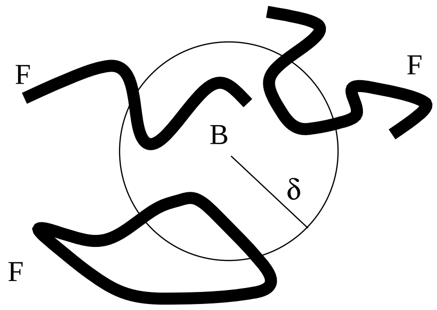


Figure 3.3: We would like to choose a point inside a ball, avoiding forbidden regions of the ball.

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 $\delta_1 = 6.31\,\varepsilon^{1/6}K^{1/3}LV^{1/3}$ and $\delta_2 = 2.15\,\varepsilon^{1/6}KLV^{1/3}$, 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 has been 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 [15], as defined in Section 3.2, are removed. The algorithm which we have chosen for vertical decomposition is the partial decomposition algorithm [47], and the list of degeneracies has been defined accordingly.

We remove the degeneracies by a perturbation of the coordinate system orientation (i.e., no geometric or topologic changes to the data are incurred). 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.

The situation here is significantly different from the situation with the inherent degeneracies. Ensuring an ε-separation does not solve any of the decomposition-induced degeneracies, since even distant entities can have close x values or close projections. We observe that all the degeneracies are derived out of the orientation of one or more geometric entities relative to an axis. Since all the decomposition-induced degeneracies are related with orientation, we choose a new coordinate system and thus instantly change the orientation of all entities without changing their geometric or topologic data. Once we have chosen a coordinate system, we need a way to measure the correctness of the new orientations. For example, we would like to make sure that according to our new coordinate system no facet is vertical (or almost vertical), but the term 'almost vertical' has to be defined. Looking over the list of decomposition-induced degeneracies, we see that they all deal with 'almost' being parallel to one of the axes, namely 'almost' having an angle of zero with one of the axes. This leads us to choosing a threshold value ω, which measures this 'almost' term. If the discussed angle is less than ω, then it is 'almost' zero. ω ensures an angular separation, which is called here *ω-separation*. We can now rephrase the degeneracies listed in Section 3.2 in a more accurate way:

vertical-facet Let f be a facet. The angle between f and the z-direction is less than ω.

equal-x Let v_1, v_2 be vertices. Let Π be a plane perpendicular to the vector $v_1 - v_2$. The angle between Π and the x-direction is less than ω.

concurrent-vertex-edge Let v and e be a vertex and a non-incident edge respectively. Let Π be a plane through v and e. The angle between Π and the z-direction is less than ω.

concurrent-intr2-edge Let i and e be an intr2 and a non-incident edge respectively. Let Π be a plane through i and e. The angle between Π and the z-direction is less than ω.

concurrent-three-edges Let e_1, e_2, e_3 be non-incident edges. Let l be a line intersecting all three edges. The angle between l and the z-direction is less than ω.

We wish to determine the biggest ω that will allow us running the coordinate system step efficiently. If we choose an ω value that is greater than the floating point resolution, the above degeneracies can be safely and consistently removed, and the vertical decomposition algorithm will run robustly.

Each of the first four degeneracies defines a collection of planes. The first one defines the collection of supporting planes of each facet f, and the following three define planes for each pair of distinct vertices, vertex and non-incident edge, or intr2 and non-incident edge respectively. Each plane defined by the first degeneracy is the locus of coordinate system z-directions that will cause the related facet to be vertical, namely have zero angular separation. In a similar way, each of the planes defined by the other three degeneracies is the locus of x or z directions that will cause another decomposition-induced degeneracy. In order to obtain a valid orientation of the coordinate system, the planes defined by *equal-x* must be ω-separated from the x-direction and the planes defined by the degeneracies *vertical-facet*, *concurrent-vertex-edge* and *concurrent-intr2-edge* must be ω-separated from the z-direction. The degeneracy of type *concurrent-three-edges* will be discussed later.

We next analyze the ω-separation for a single plane Π. The analysis is very similar to the one presented in [29, degeneracy of type II], and we repeat it here for clarity and completeness. Let S be a unit sphere whose center is located in Π. Let Π' and Π'' be two planes parallel to Π, each on a different side of Π, and such that any plane Π_0 passing through the center of S and tangent to the circle $\Pi' \cap S$ (or the circle $\Pi'' \cap S$) makes an angle ω with Π. See Figure 3.4 for an illustration. We call the portion of S between $\Pi' \cap S$ and $\Pi'' \cap S$ the *ω-strip* of Π. The area of such a strip is $4\pi \sin \omega$.

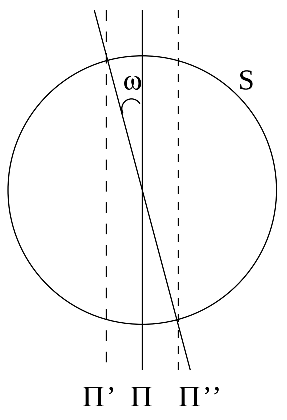


Figure 3.4: A cross section of S.

We choose the orientation of the coordinate system uniformly at random. The intersection of S with the x,y and z chosen axes induces three points that will be denoted S_x, S_y, S_z respectively. Each ω-strip defines forbidden loci for either S_x or S_z.

Recall that ω-strips are induced by planes Π, which are defined by the the first four degeneracies. The degeneracy of type *concurrent-three-edges* also defines forbidden loci on the sphere S, but these loci are more complicated and detailed in Appendix C. In short, each triple of edges defines forbidden loci for S_z, with area of $2501\pi \sin\omega$.

Let F_1, F_2, F_3, F_4, F_5 be the forbidden loci defined by the degeneracies *vertical-facet*, *equal-x*, *concurrent-vertex-edge*, *concurrent-intr2-edge*, *concurrent-three-edges* respectively. We next compute a bound for the area of each forbidden loci. Notice that we bound the number of facets by n, and therefore the number of edges and vertices is bounded by $3n$ each. The number of edges that might intersect a facet is bounded by $2K$.

1. $Area(F_1) \leq n \cdot 4\pi \sin\omega$.
2. $Area(F_2) \leq \binom{3n}{2} \cdot 4\pi \sin\omega$.
3. $Area(F_3) \leq 3n \cdot 3n \cdot 4\pi \sin\omega$.
4. $Area(F_4) \leq (n \cdot 2K) \cdot 3n \cdot 4\pi \sin\omega$.
5. $Area(F_5) \leq n^3 \cdot 2501\pi \sin\omega$.

We would like to have probability of more than $\frac{1}{2}$ for choosing a valid orientation. Let $Prob(W)$ denote the probability of occasion W. Let W_x and W_z denote the occasions of choosing invalid x and z directions respectively. We wish to have $Prob(W_x \text{ or } W_z) < \frac{1}{2}$. The following inequality is obtained assuming $n \geq 10$.

$$\begin{aligned} Prob(W_x \text{ or } W_z) &\leq Prob(W_x) + Prob(W_z) \\ &\leq \frac{Area(F_2)}{Area(S)} + \frac{Area(F_1) + Area(F_3) + Area(F_4) + Area(F_5)}{Area(S)} \\ &\leq \frac{(626.61 + 0.6K)n^3 \sin\omega \cdot 4\pi}{4\pi} \end{aligned}$$

The bound obtained for ω is therefore:

$$\sin\omega < \frac{1}{(313.31 + 0.3K)n^3}$$

Chapter 4

Complexity Analysis

In the main theorem there is a part that summarizes the complexity of our scheme. In this chapter we prove it and add more information concerning the performance of our algorithm.

4.1 Time Complexity

Recall that L is the maximum length of an edge in the input data, δ_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. 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 4.1(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 have possibly been perturbed by at most δ_1, and then we add 2ε 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 4.1(b)).

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 2.3. See Figure 4.1(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.

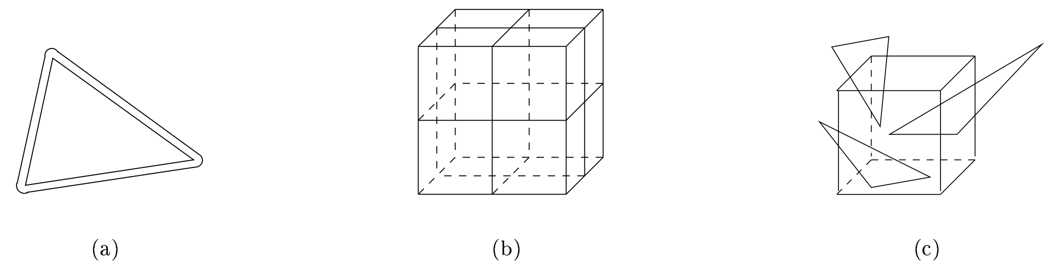


Figure 4.1: (a) A two-dimensional illustration of an expanded facet. (b) An expanded entity can intersect at most 8 grid cubes. (c) We assume that the number of expanded facets intersecting a single grid cube is bounded by a constant D.

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.

4.1.1 Local Step

Together with the choice of δ_1 (as described in Section 3.6), the guaranteed expected time of this step is $O(nD)$, as detailed below.

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.

4.1.2 Global Step

Together with the choice of δ_2 (as described in Section 3.6), the guaranteed expected time of this step is $O(n \log^3 n + nDK^2)$, as detailed below.

Partitioning Sub-Step The 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. 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 $\bar{T}$ (tested in $O(1)$ time). (ii) The whole terrain $\bar{T}$ 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 $\bar{T}$ (tested in $O(\log^3 n)$ time by [12] for ray shooting).

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 two facets 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 two facets need to be checked.

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

4.1.3 Coordinate System Step

The operation that has the highest complexity in this step is the removal of degeneracy of type *concurrent-three-edges*, i.e, the detection of triples of edges whose xy-projections intersect in three points that are too close [overlap]. Since every triple of edges need to be checked, we get a time complexity of $O(n^3)$.

For clarity and completeness we give here the complexity of the removal of the rest of the decomposition-included degeneracies.

vertical-facet Every facet needs to be checked, so we obtain a time complexity of $O(n)$.

equal-x Every combination of two vertices needs to be checked, so we obtain a time complexity of $O(n^2)$.

concurrent-vertex-edge Every combination of a vertex and an edge needs to be checked, so we obtain a time complexity of $O(n^2)$.

concurrent-intr2-edge Every combination of an intr2 and an edge needs to be checked, so we obtain a time complexity of $O(n^2K)$.

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

4.2.1 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.

4.2.2 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 as mentioned in Section 4.1; see [12]. 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 $O(n \log n + nK^2)$.

Chapter 5

Swept volume

The motivation for our work is the swept volume application. In this chapter we define swept volumes, explain why they are important, and describe the reasons that made us develop our perturbation scheme. We also describe the swept volume algorithm that we have implemented, and tell about the way in which it uses our perturbation scheme.

5.1 Definitions

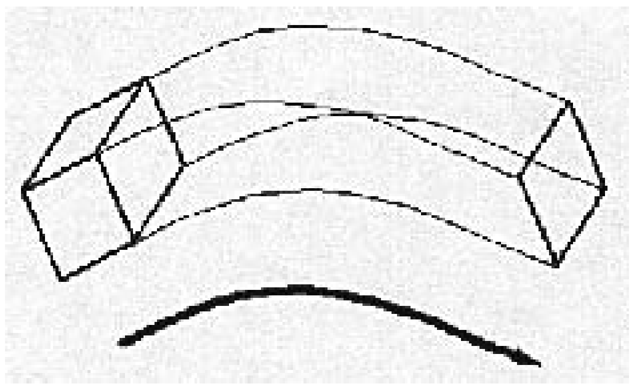


Figure 5.1: Swept volume of a cube

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 also called a *generator*. The motion of the generator is called a *sweep*. See Figure 5.1.

Let A denote a generator which is swept along some trajectory. Let A_t represent an instance of A during the sweep over a parametrized unit interval t∈I = [0,1], where t=0 and t=1 represent the instances of A at the initial and final locations along the trajectory, respectively. So the swept volume of A is the union of A_t, for all t∈I:

$$SV(A) = \bigcup_{t \in I} A_t$$

Unfortunately, this definition does not give a direct way to generate swept volumes.

5.2 Practical Uses

Swept volumes play an important role in many CAD/CAM applications including geometric modeling, robot motion planning, numerical control cutter path generation, and assembly planning.

Geometric Modeling Translational and rotational swept surfaces are used in CAD systems. The best known are surfaces of revolution [13]. More complex geometry can be generated by using higher order sweep trajectories and generating surfaces.

Robot Motion Planning Swept volumes can be used to evaluate safe paths in robot motion planning. Motion verification can be done by determining whether a robot moving along a prescribed path collides with obstacles. Collision free path finding can be done in many ways [22],[34]. Two of them are:

Heuristic search A path is hypothesized and checked. If collision is incurred by that path, an alternative path is suggested using some heuristic rules. Collision is checked again for the new path, and the procedure is repeated until a collision free path is found. The collision checks can be carried out by first computing the swept volume.

Explicit spatial planning approach The goal is to plan the motion of a robot say in a 2-dimensional work space that has some obstacles in it. For translational movements, we compute the space that is free of obstacles for the robot by shrinking the robot to a point and correspondingly growing the work space obstacles to represent the configuration space obstacles. The resulting configuration space is still 2-dimensional. When rotations of the robot are allowed, the configuration space is 3-dimensional $\mathbb{R}^2 \times [0 : 2\pi)$. We take the 2-dimensional configuration space which has been computed for translational movements, and extend it to a 3-dimensional one by continuously rotating the one copy of every translational configuration obstacle [16, Chapter 13],[39, Chapter 8]. See Figure 5.2. The creation of these 3-dimensional configuration obstacles can by done by using swept volumes.

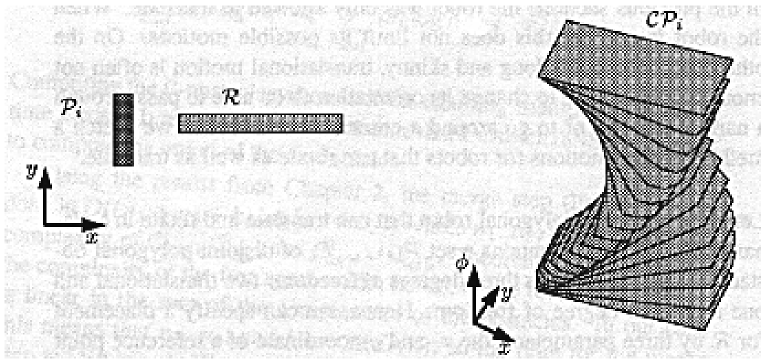


Figure 5.2: An obstacle in the configuration space of a rotating and translating robot

Assembly Planning Swept volumes are applied to resolve maintainability issues that arise during the design of complex mechanical systems. The designer needs to be able to create and visualize the accessibility and removability of individual components of the system. While removing an object from the assembly, there is the need for a safe and feasible trajectory. A safe trajectory is one that a component can follow without colliding with other components of the assembly. A feasible trajectory is often defined to be a trajectory that can be performed by a human [27],[34].

Numerical Control Path Planning Swept volumes are used to show the removal of material by a tool [51]. Collision with other obstacles can be detected, and one can find out whether a workpiece after being machined has the same shape and dimensions as prescribed [8],[35].

5.3 Related Research

Research on swept volume generation has received considerable attention over the last decade.

Restricting the motion to translational only is dealt by Leu, Park and Wang [36].

Wang and Wang [51] present a solution that uses a 3D z-buffer to compute a family of critical curves from a moving solid. They restrict the generating object to a convex set. Their approach is driven by the need to compute numerical control tool paths.

Weld and Leu [52] discuss the generation of a swept volume by a polyhedral object in the three-dimensional case. They show that the representation of a swept volume in $\mathbb{R}^n$ generated from an n-dimensional object can be reduced to developing a geometric representation from the swept volume of its $(n-1)$-dimensional boundary. In other words the volume swept by a moving polyhedral surface can be computed by sweeping the boundary elements (polygonal facets) of the polyhedral surface together with the polyhedral surface itself at a finite set of specific positions during the sweep. They analyze the volume swept by a polygon in three-dimensional space and show that this volume is bounded by copies of the polygon at its initial and final positions, ruled surfaces generated by sweeping of the polygon edges, and in special cases, volume swept by the interior points of the polygon (called developable surfaces). They observe that those special cases occur when successive instances of the polygon at times t and $t + \varepsilon'$ intersect each other (for $\varepsilon' > 0$).

Martin and Stephenson [37] use envelope theory to generate the swept volumes of three-dimensional objects. They provide a general solution, but for complicated sweeps this solution may take a prohibitive amount of computer time.

In two-dimensional space, Sambandan, Kedem and Wang [43] develop a way for choosing the candidate vertices and edges which eventually take part in the generated swept area. This technique has been used in later research on 3D swept volume [32].

Blackmore and Leu and then Wang [5],[6],[7] present a differential equation method which produces exact geometry of the swept volume.

An implicit modeling approach to swept volume generation is given by Schroeder, Lorensen, and Linthicum [45]. They generate an implicit model by assigning a distance value from each voxel to the surface of the object. Then the implicit model is swept through the workspace by sampling it as it is transformed along the sweep trajectory. Finally the swept surface is generated using the iso-surface extraction algorithm marching cubes.

Abrams and Allen [2],[3] continue the topological approach of Weld and Leu. They use facet sweeping as well, but they have added efficiency and robustness elements to it: They approximate both the ruled surfaces generated by edge sweeping, and the developable surfaces. Yet, they encounter robustness problems with arrangement computations.

Hu and Ling [32] use envelope theory for generating swept volumes of quadric surfaces. They describe motion using the instantaneous screw-axis presentation. In their computations, they perform the sweep only on facets which are candidates for contributing to the swept volume boundary.

Abel-Malek and Yeh [1] describe the boundary of swept geometric entities of multiple parameters. They define a constraint function and impose a rank-deficiency condition on the constraint Jacobian of the sweep to determine singular sets.

Xavier [53] presents a fast, implemented swept volume technique, aimed mostly for collision detection. Among others, he uses polyhedral approximations of convex hulls.

5.4 Implementation

For our swept volume application we have chosen to implement the algorithm given by Abrams and Allen [3], 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 [3] 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 [15], extended to the case of polyhedral surfaces. This algorithm requires degeneracy-free input, hence before using it we apply our perturbation package.

We present here a robust version of the swept volume algorithm, based on the algorithm given by Abrams and Allen at [3]. The algorithm ignores voids, since they are of no interest to most of the practical uses of swept volumes (some of the swept volume papers explicitly say that they ignore voids). The following algorithm computes the swept volume $S(P, T)$ of a polyhedral surface P and a trajectory T.

1. For each of the n Polygons p_i of P, do the following:

(a) Step each edge of p through the trajectory T using any step size Δt, connecting adjacent copies of each edge by forming triangles. Call this set of triangles $\mathcal{F}$.

(b) Add a copy of p at its initial and final positions to $\mathcal{F}$.

(c) Run the Sliding Motion Test (as detailed in [3]) on p_i, using the same step size Δt. Add all of the facets generated by this algorithm to $\mathcal{F}$.

(d) The set $\mathcal{F}$ is now a superset of the boundary of the actual volume swept. Apply the **robustness scheme** on set $\mathcal{F}$ and then, using **vertical decomposition**, compute the set $\mathcal{A} = \mathit{Arrangement}(\mathcal{F})$.

(e) Traverse the boundary of the infinite cell, i.e., the outer-most boundary of $\mathcal{A}$. This is V_i, or $S(p_i, T)$.

2. Compute $V = \bigcup_{i=1}^{n} V_i$.

We have implemented this algorithm, and indeed, we have not encountered robustness problems when using it.

Chapter 6

Experimental Results

We have implemented our scheme and proven it in practice. In this chapter we present our 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 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 6.1, 6.2 and 6.4 can be found in [41]. Many of the software classes used in our implementation belong to the CGAL[1] library.

In Table 6.1 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 effect on $\bar{d}$ is striking. The swept volume of 302 facets is depicted in Figure 2.1.

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 6.1: 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.

In Table 6.2 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 6.3 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. 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).

[1]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 6.2: 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 6.3: 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.

In Table 6.4 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 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 6.1) 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 6.2. 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.

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 6.4: 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.

K and D are defined as the maximum values during the perturbation running time. In order to simplify certain coding and debugging procedures, the $K, \bar{k}, D$ and $\bar{d}$ values given here are the static values that have been obtained before the perturbation has been run. As explain in Section 2.3, those values are good enough, since δ is very small and therefore the number of facets intersecting a single facet (for $K, \bar{k}$) and the number of expanded facets intersecting a single cube grid (for $D, \bar{d}$) hardly change during running time.

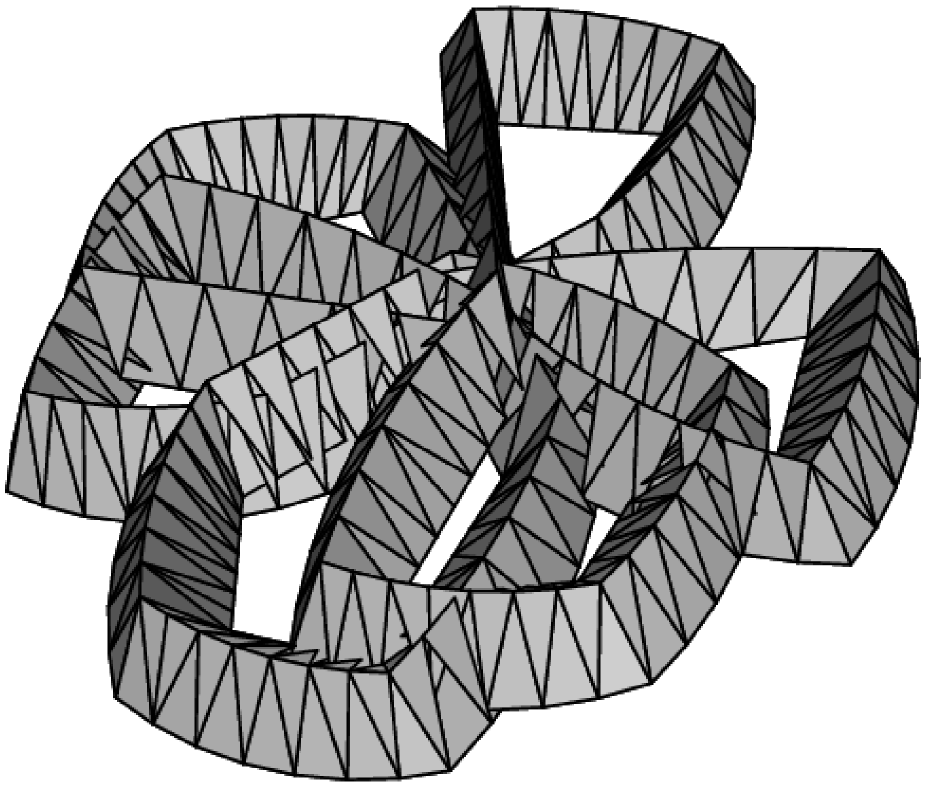


Figure 6.1: A swept volume inducing many degeneracies. The dense intersection area in the center makes it difficult to find a valid perturbation.

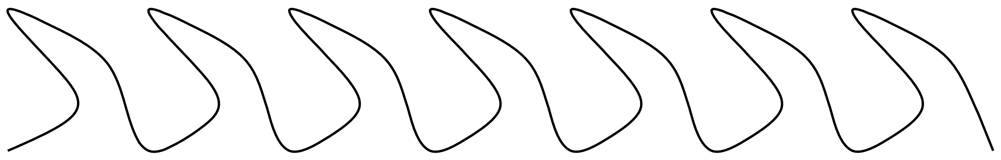


Figure 6.2: A one dimensional illustration of a swept volume, where the xy-partitioning process results in many terrains and therefore many connectors.

Chapter 7

Improvement

In this chapter we suggest an improvement over the perturbation scheme that has been presented so far.

Why Do We Need an Improvement? We say that facet f_1 is *influenced* by facet f_2 if f_2 affects the perturbation distance of f_1. The obvious ways for influencing a facet are by intersecting it or being ε 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 influenced only by the facets that intersect it, i.e., we assume that the number of facets influencing a certain facet is bounded by K (recall that K bounds the number of facets intersecting a single fact).

This assumption is good enough for practical cases, i.e., real world input data, and makes the exposition of our ideas simpler. However, since we want our scheme to cover also theoretical cases, we now turn to relaxing this assumption.

We do not make the improvement an integral part of the perturbation scheme, since it is needed only for non realistic inputs, whereas this thesis is motivated by real world (and mostly industrial) inputs. We present the improvement in order to show that we have a solution even for the most difficult theoretical inputs.

Example Here is an example in which the maximum number of facets influencing a single facet is greater than K.

Suppose that every planar layer in Figure 7.1 is an arrangements of 100x100 planar triangles that are 1.5ε away from each other. Suppose the input data consists of 100 such arrangements, ordered vertically one on top of the other, with a space of 1.5ε between every two layers. So far we described a perfectly robust arrangement of 100^3 triangles. Once this arrangement is given as input to our scheme, there will not be a single change in it. Now suppose the $100^3 + 1st$ triangle of the input data is a horizontal triangle that is located in the center of the arrangement of 100^3 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 distance that assures a valid placement is outside the arrangement of 100^3 triangles. In other words, the number of triangles influencing the last triangle is not bounded by K.

Solution Let Ψ be the maximum number of facets influencing 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. Ψ is the maximum $\psi(f)$ over all facets f in the collection P of input polyhedral surfaces.

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

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

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

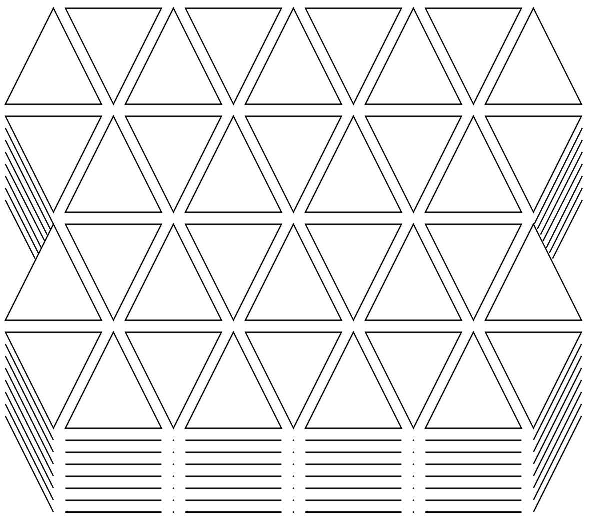


Figure 7.1: An arrangement of triangles where the maximum number of facets influencing a single facet is greater than K.

Chapter 8

Conclusion

We have presented a perturbation scheme for a collection of polyhedral surfaces in 3D space. The scheme overcomes degeneracies and precision problems. It is useful for applications that use such polyhedral surfaces, allowing them the usage of finite precision arithmetic and saving them the need of handling degenerate cases. Our scheme takes [29] one step ahead, extending it from spheres into 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 degeneracies derived out of the fact that polyhedral surfaces may have arbitrarily many degrees of freedom.

The key idea of our scheme is that for a given resolution parameter $\varepsilon > 0$ we slightly perturb the input data such that features of the three-dimensional arrangement of the polyhedral surfaces are at least ε apart. We have presented a tradeoff between the magnitude of the perturbation and the expected running time of the scheme, and have shown that using a smaller perturbation size (which is a favored option) may result in larger expected running time.

We have presented experimental results obtained while using standard floating point arithmetic and introduced fairly efficient running time. The algorithm that we have presented is simple and easy to program.

The initial motivation for developing the perturbation scheme is a swept volume application, which employs vertical decomposition as its final step. The vertical decomposition algorithm allows no degenerate input, and we have created our perturbation scheme in order to supply the vertical decomposition algorithm a degeneracy-free input.

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

We propose several directions for future research:

1. Extend the perturbation scheme to non-manifold polyhedral objects, such as polyhedral 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). We conjecture that the optimization problem is NP-hard, in which case we are interested in an approximation to the optimization.
3. Prove that δ is bounded also for the case mentioned in Section 3.5, where more than two terrains meet at a point. We remind the reader that such a case works well in practice, but has not been proven yet in theory.

Appendix A

Inherent Degeneracies

In this appendix we give the full list of inherent degeneracies, and explain why most of the degeneracies are special cases of others.

A.1 Full List of Inherent 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 overlapping 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 overlapping 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 [overlaps] a non-incident intr3.

A.2 Special Cases

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

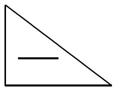
vertex-vertex is a special case of degeneracy of type *vertex-facet*.

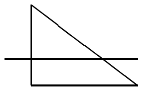
vertex-edge is a special case of degeneracy of type *vertex-facet*.

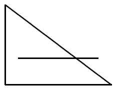
vertex-segment is a special case of degeneracy of type *vertex-facet*.

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

edge-facet is divided to the following cases:

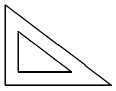
 is a special case of degeneracy of type *vertex-facet*.

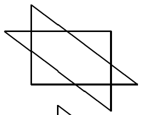
 is a special case of degeneracy of type *edge-edge*.

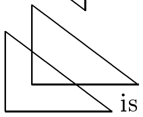
 is a special case of both degeneracy of type *vertex-facet* and *edge-edge*.

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

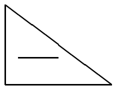
facet-facet is divided to the following cases:

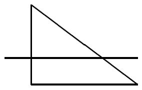
 is a special case of degeneracy of type *vertex-facet*.

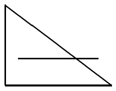
 is a special case of degeneracy of type *edge-edge*.

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

facet-segment is divided to 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 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*.

Appendix B

Bounding the Size of δ

In this appendix we give more details on the shape and volume of forbidden loci that induce degeneracies. The size of δ 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.

B.1 Preliminaries

In this section we describe our notation and certain preliminary calculations which are related to computing forbidden loci.

B.1.1 A Brief Reminder

B.1.1.1 General

$\boldsymbol{K}$ denotes the maximum number of facets intersecting any single facet

$\boldsymbol{L}$ denotes the maximum edge length

$\boldsymbol{V}$ denotes the maximum number of vertices in one polyhedral surface

$\boldsymbol{\delta_1, \delta_2}$ denote the maximum perturbation radi for the local and global steps respectively.

$\boldsymbol{\delta}$ is defined as the sum of δ_1 and δ_2.

B.1.1.2 Local Step

We remind the reader that we remove the local inherent 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.

Recall that $P = \{P_1, P_2, \ldots, P_m\}$ is a collection of m (possibly intersecting) polyhedral surfaces and that s_i is the number of vertices in P_i, $1 \leq i \leq m$. Recall that $v_1, v_2, \ldots, v_{s_i}$ is an ordering of the vertices in P_i. Recall that Q_r denotes the data structure that was generated while processing vertices $v_1, v_2, \ldots, v_r$, $1 \leq r \leq s_i$. 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 $\{v_1, v_2, \ldots, v_r\}$.

B.1.1.3 Global Step

We remove the global inherent 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.

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

B.1.2 Definitions of Symbols

$\boldsymbol{\mathcal{L} := L + 2\delta_1}$ denotes the maximum edge length after perturbing both endpoints.

$\boldsymbol{F_{loc\text{–}type1\text{–}type2}}$, $\boldsymbol{F_{loc\text{–}type1\text{–}type2\text{–}subcase}}$, $\boldsymbol{F_{glob\text{–}type1\text{–}type2\text{–}0}}$, $\boldsymbol{F_{glob\text{–}type1\text{–}type2\text{–}0\text{–}subcase}}$ denote the set of forbidden loci for a certain degeneracy. The initial *loc/glob* denotes whether a local/global degeneracy is being dealt with. The loci is for the removal of degeneracy of type $type1 - type2$ where v, e, f, s, i represent $vertex, edge, facet, segment, intr3$ respectively. Then the case which is dealt is specified (0 in the example here), and if there is a sub-case, then its number is also specified.

For local degeneracies, $F_{loc\text{–}type1\text{–}type2}$ and $F_{loc\text{–}type1\text{–}type2\text{–}subcase}$ denote the volume which is forbidden for placements of the currently added **vertex** v_r. For global degeneracies, $F_{glob\text{–}type1\text{–}type2\text{–}0}$ and $F_{glob\text{–}type1\text{–}type2\text{–}0\text{–}subcase}$ denote the forbidden **translation vector** of the currently added terrain T_j.

$\boldsymbol{S_1 \oplus S_2}$ denotes the Minkowski sum of sets S_1 and S_2. Similarly, $S_1 \ominus S_2$ denotes the Minkowski sum of S_1 and $-S_2$, where $-S_2 := \{-q \mid q \in S_2\}$. Recall that the Minkowski sum of two sets of points S_1 and S_2, denoted $S_1 \oplus S_2$, is defined as $S_1 \oplus S_2 := \{p + q \mid p \in S_1, q \in S_2\}$.

$\boldsymbol{Set(type, object)}$ denotes the set of geometric entities of type *type* which are contained in *object*. For example, $Set(vertex, M_{j-1})$ is the set of vertices in M_{j-1}, and $Set(intr3, T_j)$ is the set if intr3 in T_j.

$\boldsymbol{Volume(entity)}$ denotes the volume of a three-dimensional geometric entity.

$\boldsymbol{Area(entity)}$ denotes the area of a two-dimensional geometric entity.

$\boldsymbol{MaxInter(ent_1, ent_2)}$ denotes the maximum number of entities of type ent_1 that entities of type ent_2 might intersect. Since we are dealing with floating point arithmetic, $MaxInter(ent_1, ent_2)$ can be interpreted as the maximum number of entities of type ent_1 that might be too close to entities of type ent_2. The possible values are shown in Table B.1. The empty squares in the table are for values which are not used in volume computation. Notice that some of the values are influenced by the order of degeneracy removal. For example, $MaxInter(edge, edge)$ could have been $3K$ for the case where an edge intersects a vertex and an opposite edge of a triangle, but it is $2K$ because this check is being done after degeneracies where an edge is close to a vertex are removed.

ent_2 / ent_1	facet	edge	vertex
facet	K		K
edge		$2K$	$2K$
vertex	$3K$	$2K$	
segment		$\binom{K}{2}$	
intr2	$2\binom{K}{2}$		
intr3	$\binom{K}{3}$	$\binom{K}{3}$	

Table B.1: $MaxInter(ent_1, ent_2)$ denotes the maximum number of ent_1 that ent_2 might intersect.

$\boldsymbol{N}$ denotes the maximum number of vertices on the boundary of a polyhedral terrain. Then N is also the maximum number of edges on the boundary of a polyhedral terrain.

B.1.2.1 Symbols and Lengths Related to Connectors

In the global step, polyhedral surfaces are being split into a number of terrains, and all edges and vertices that belong to the boundary of terrains are duplicated.

Let e be a boundary edge in M_{j-1}. Let e' be its instance in T_j. Let v and u be the vertices of e. Let their instances in T_j be v' and u', respectively. After the perturbation of T_j, e and e', v and v', and u and u' no longer overlap. Let H be the connector created between e and e'. H is composed out of two triangles. See Figure B.1.

The maximum length of the edges (v, v') and (u, u') is δ_2, and therefore the maximum length of the diagonal edge (u, v') is $\mathcal{L} + \delta_2$.

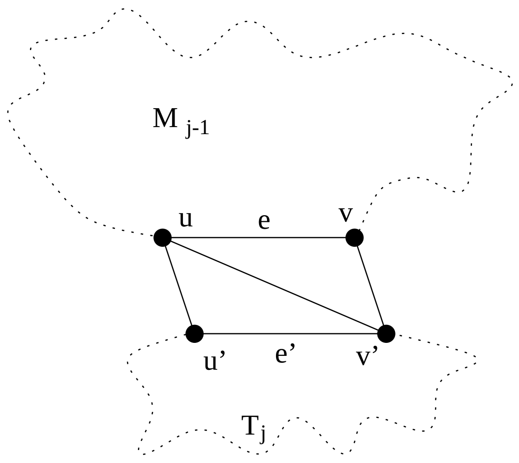


Figure B.1: A typical connector.

B.1.2.2 Expanded Shapes

Let the term *μ-expanded shape* denote the Minkowski sum of a planar polygon/edge/vertex and a 3D ball $B(0, \mu)$, where μ is a positive real number. For polygonal shapes, the maximum edge length of each is $\mathcal{L}$.

Let $Exp(\mu, S)$ denote the μ-expansion around the shape S.

We compute here the volume of a variety of μ-expanded shapes.

μ-expanded edge (or $Exp(\mu, edge)$)

$$Volume = \mathcal{L}\pi\mu^2 + \frac{4}{3}\pi\mu^3 = \pi\mu^2(\mathcal{L} + \frac{4}{3}\mu)$$

The first term in the sum is the volume of a cylinder. The second term in the sum is the volume of two half balls, one at each endpoint of the edge.

μ-expanded triangle (or $Exp(\mu, triangle)$)

$$\begin{aligned} Volume &= 2\mu\frac{1}{2}\mathcal{L}^2 + 3 \cdot \frac{1}{2}\mathcal{L}\pi\mu^2 + 2\frac{1}{2} \cdot \frac{4}{3}\pi\mu^3 \\ &= \mu(\mathcal{L}^2 + \frac{3}{2}\pi\mathcal{L}\mu + \frac{10}{3}\pi\mu^2) \end{aligned}$$

The first term in the sum is the maximum volume of a triangular prism. The second term in the sum is the volume of three half cylinders. The third term in the sum is the volume of $2\frac{1}{2}$ balls in the corners. (There are three balls, but the volume of half a ball should be subtracted since it is internal to the triangular prism).

μ-expanded parallelogram (or $Exp(\mu, parallelogram)$)

$$\begin{aligned} Volume &= 2\mu\mathcal{L}^2 + 4 \cdot \frac{1}{2}\mathcal{L}\pi\mu^2 + 3 \cdot \frac{4}{3}\pi\mu^3 \\ &= 2\mu(\mathcal{L}^2 + \pi\mathcal{L}\mu + 2\pi\mu^2) \end{aligned}$$

The first term in the sum is the maximum volume of a parallelogram prism. The second term in the sum is the volume of four half cylinders. The third term in the sum is the volume 3 balls in the corners. (There are 4 balls, but the equivalent of of a single ball is internal to the prism).

B.1.3 Infrastructure Resolution Parameters: ρ and λ

The invariants defined in Sections 3.4 and 3.5 require that every pair of geometric entities is at least ε-away from each other. In this section we tighten the requirements.

During our degeneracy removal procedure, we use more crude resolution parameters for specific pairs of entities. The additional resolution parameters are named ρ and λ. The situations where ρ and λ distances are ensured, are where we create an infrastructure for a safe removal of other degeneracies. The sections that deal with ensuring infrastructure distance explicitly mention that in their headers.

Let ρ be a positive real number such that $\rho > \varepsilon$. Both ρ and $\frac{\varepsilon}{\rho}$ should be very small.

Let λ be a positive real number such that $\lambda > \rho$. Both λ and $\frac{\rho}{\lambda}$ should be very small.

Next we explain the situation that raises the need for ρ. In Figure B.3 there are two sectors of a circle. We tell more about these sectors in Section B.1.4.2, and here we treat them as granted. For both sectors we need to prove that the upper bound of their area is very small. An area is considered 'very small' if it is influenced by ε or some other small resolution parameter.

The upper bound for the area of the sectors in Figure B.3 is achieved when the angle α reaches its upper bound. We show in Section B.1.4.2 that $\sin\alpha = \frac{\varepsilon}{d(v,w)}$ for the sector in Figure B.3(a) and that $\sin\alpha = \frac{\varepsilon}{d(v',w)}$ for the sector in Figure B.3(b). This means α becomes larger as $d(v,w)$ or $d(v',w)$ become smaller. According to our robustness requirements, $d(v,w) > \varepsilon$ and $d(v',w) > \varepsilon$, and therefore in both cases $\sin\alpha < \frac{\varepsilon}{\varepsilon}$, which means α is bounded by 90^o. Thus the sector is bounded by half a circle, the area of which is not influenced by ε nor any other resolution parameter, and we cannot claim that such an area is very small.

We solve this problem in the following way: Whenever two entities ent_1, ent_2 participate in the formula $\sin\alpha = \frac{\varepsilon}{d(ent_1,ent_2)}$, we make sure that the distance between those entities is greater than ρ. Therefore we get that $\sin\alpha < \frac{\varepsilon}{\rho}$, which causes the area of the sector to be influenced by $\frac{\varepsilon}{\rho}$ and therefore be very small.

The need for λ arises because of similar reasons, where $\sin\alpha = \frac{\rho}{d(ent_1,ent_2)}$ and thus we make sure that $d(ent_1, ent_2) > \lambda$.

The actual values of ρ and λ are determined only at the final stage of the computation.

Table B.2 shows which entities should be more than ε-away from each other (namely ρ-away or λ-away from each other). In addition it points to the degeneracy removal procedures that require those extra terms.

Requirement		Used	
Term	Fulfilled in Section	For removal of degeneracy	In Section
$d(v,e) > \rho$ where v, e are in different terrains and e is a boundary edge	B.3.1.2	*vertex-facet* (global, special case)	B.3.1.4
		edge-edge (global, special case)	B.3.2.2
		edge-segment (global, special case)	B.3.3.2
		edge-segment (global, special case)	B.3.3.3
		edge-segment (global, special case)	B.3.3.5
		facet-intr3 (global, special case)	B.3.4.2
		facet-intr3 (global, special case)	B.3.4.3
$d(v,f) > \rho$ where v, f are in different terrains and v is a boundary vertex	B.3.1.3	*edge-segment* (global, special case)	B.3.3.6
$d(v,e) > \rho$ where v, e are in the same surface	B.2.2	*vertex-facet* (local)	B.2.3
		edge-edge (local)	B.2.4
		vertex-facet (global, special case)	B.3.1.5
		edge-edge (global, special case)	B.3.2.3
		edge-segment (global, special case)	B.3.3.4
$d(v_1,v_2) > \lambda$ where v_1, v_2 are in the same surface	B.2.1	*vertex-edge* (local)	B.2.2

Table B.2: The list of entities that should be more than ρ- or λ-away from each other, and pointers to the procedures where those extra terms are used. The symbols v, e, f denote vertex, edge and facet respectively.

B.1.4 Basic Computations

The computations in this section are fundamental and used in several places later on.

B.1.4.1 Bounding the Number of Edges and Facets

According to Euler's formula applied to the case of possibly no outer face, $v + f - e \geq 1$, and since every facet is triangular, so $e \geq \frac{3}{2} \cdot f$, we get:

- The maximum number of facets in a polyhedral surface is $2V - 2$.
- The maximum number of edges in a polyhedral surface is $3V - 3$.

Notice that these values are also bounds for polyhedral terrains.

B.1.4.2 Bounding the Volume of a Sphere Slice

Let v_{static} be a vertex whose location has already been determined. Let v_{dynaic} be a vertex that shares an edge $(v_{static}, v_{dynamic})$ with v_{static}, and whose location has not yet been determined.

The situation as described above can happen both in the local and the global steps:

Local step v_{dynaic} is the vertex that is currently being added to the data structure Q_r. (v_{dynaic} is denoted as v_r in Section 3.4). v_{dynaic} induces an edge $(v_{static}, v_{dynamic})$ where v_{static} has already been located in a former stage.

Global step v_{dynaic} is a boundary vertex of a polyhedral terrain T_j that is currently being added to the data structure M_j. v_{static} is the vertex from which v_{dynaic} has been split, and belongs to a terrain that has already been located in a former stage. v_{dynaic} creates an edge $(v_{static}, v_{dynamic})$ in the connector $con(T_j)$. In the symbols defined in Section B.1.2.1, v_{static} is equivalent to v and v_{dynaic} is equivalent to v'.

Let len denote the maximum possible length of the edge $(v_{static}, v_{dynamic})$. For the local step, it is the maximum length of an edge, i.e., $\mathcal{L}$. For the global step, it is δ_2 for a connector edge that connects two vertices that were formerly split (like (v, v') in Figure B.1) and $\mathcal{L} + \delta_2$ for a connector diagonal (like (u, v') in Figure B.1).

The full range for perturbing v_{dynaic} is the sphere $B(v_{static}, len)$. In our computations, we try to identify the sub volume of this sphere, in which locating v_{dynaic} induces a degeneracy.

Usually we bound this 'forbidden' volume in a sphere slice. In Figure B.2 such a slice is shown for the parameters $v_{static} = v$, $v_{dynamic} = v'$, $len = \delta_2$, and therefore fits the creation of a connector in the global step, with symbols as defined in Section B.1.2.1. Notice that v_{dynaic} can be located anywhere inside the sphere slice.

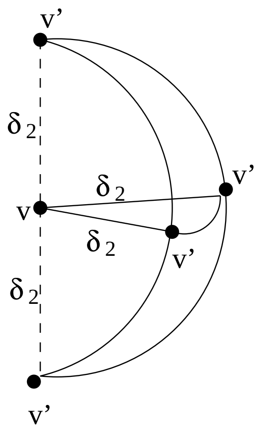


Figure B.2: A slice of a perturbation range sphere. $v_{static} = v$, $v_{dynamic} = v'$, $len = \delta_2$.

Let the term *axis* denote the sphere pole that connects the vertical facets of the slice.

Our goal is to compute the volume of the slice. Let 2α denote the angle of the sphere . (We use 2α for reasons that will become clear later).

The usual way of calculating a volume of a sphere slice is $\frac{2\alpha}{2\pi} \cdot \frac{4}{3}\pi \cdot len^3$. The way of obtaining α is by looking at the center cross section of the sphere slice (drawn inside Figure B.2), which is a disc sector.

Let w be a point in the plane of this sector. w belongs to an entity whose location has already been set (the type of that entity changes according to the degeneracy which is being dealt with), and we want w and $(v_{static}, v_{dynamic})$ to be at least ε away from each other. The sector is the union of potential locations of v_{dynaic}, where $d(w, (v_{static}, v_{dynamic})) \leq \varepsilon$. In other words, the sector is the union of potential locations of v_{dynaic}, where $B(w, \varepsilon)$ (in two dimensions) and $(v_{static}, v_{dynamic})$ intersect.

Let $Slice(w, v_{static}, v_{dynamic}, len)$ denote the sphere slice of forbidden loci for $v_{dynamic}$, where v_{static} is the vertex that has already been located, w is a geometric entity that determines the slice angle, and len is the radius of the sphere.

There are two cases when computing the angle of the sector:

1. The location of w is already set, i.e. it cannot be moved. In addition $d(w, v_{static}) > \rho$.

2. The location of w has not yet been determined, and w is rigidly attached to v_{dynaic} (i.e. w moves whenever v_{dynaic} moves and the distance and direction between them is constant). In addition $d(w, v_{dynamic}) > \rho$.

Using the sector as it is induces a rather unwieldy formula, involving sin and arcsin operations. In order to simplify the calculations we have bounded the sector by a triangle. Let the the length of the new created edge be $2b$. See Figure B.3, drawn such that $v_{static} = v$, $v_{dynamic} = v'$, $len = \delta_2$.

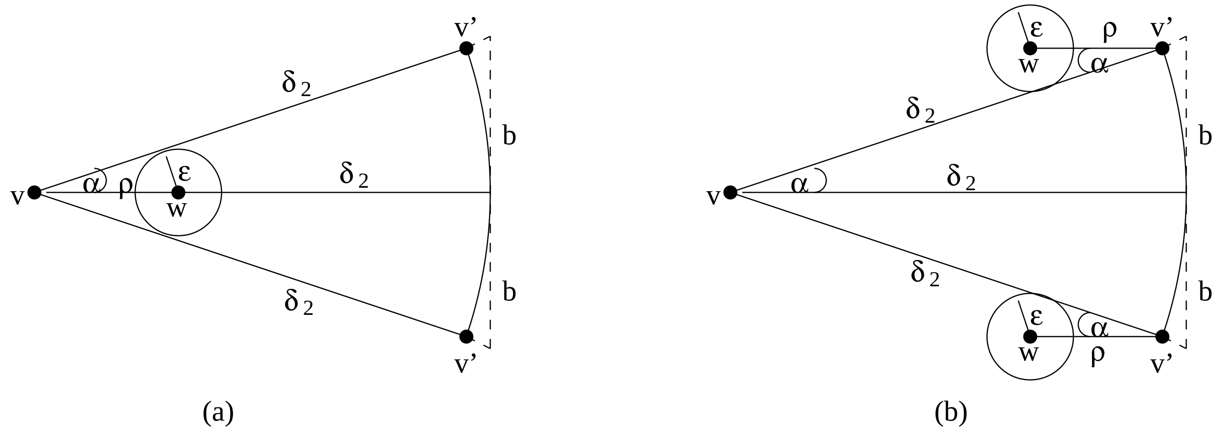


Figure B.3: A cross section of $Slice(w, v, v', \delta_2)$. (a) Smallest sector for case 1. (b) Smallest sector for case 2.

In both cases we get the smallest possible sector if the circle $B(w, \varepsilon)$ is tangent to the sector. The tangency is from the inside in case 1 (Figure B.3(a)) and from the outside in case 2 (Figure B.3(b)).

Our worst-case for volume is when α reaches its maximum possible value.

In case 1 $\sin\alpha = \frac{\varepsilon}{d(v_{static}, w)}$ and in case 2 $\sin\alpha = \frac{\varepsilon}{d(v_{dynamic}, w)}$. This means that α becomes larger as $d(v_{static}, w)$ (in case 1) or $d(v_{dynamic}, w)$ (in case 2) become smaller.

The minimum value for both $d(v_{static}, w)$ and $d(v_{dynamic}, w)$ is ρ, so we use ρ in order to find out what is the worst-case.

$$\sin\alpha = \frac{b}{\sqrt{len^2 + b^2}} \leq \frac{\varepsilon}{\rho} \Longrightarrow b^2 \leq \frac{\varepsilon^2}{\rho^2 - \varepsilon^2} len^2$$

Let $\triangle$ denote the bounding triangle of the sector.

$$Area(\triangle) = 2 \cdot \frac{b \cdot len}{2} \leq \frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}} len^2$$

Hence a bound on the ratio between the area of the sector and the area of a circle of the same radius is:

$$\frac{2\alpha}{2\pi} = \frac{Area(sector)}{\pi \cdot len^2} \leq \frac{Area(\triangle)}{\pi \cdot len^2} \leq \frac{1}{\pi} \frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}$$

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

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

B.2 Computing Forbidden Volumes for the Local Step

Suppose the incremental procedure of the local step has been carried out successfully for the first $r-1$ stages. The incremental procedure guarantees that there are no local inherent degeneracies of types *vertex-facet* and *edge-edge* in Q_{r-1}. We would like to find a location for v_r so that no local degeneracy of type *vertex-facet* or *edge-edge* is induced.

Notice that in some of the computations we need to use the maximum degree of a vertex, for example, when considering all edges incident to the vertex v_r. In such cases we choose a crude bound, and use the maximum number of edges in a polyhedral surface (when considering edges which are incident to v_r) or the maximum number of facets in a polyhedral surface (when considering facets which are incident to v_r).

Recall that sections which ensure ρ or λ distances explicitly mention the word 'infrastructure' in their headers.

B.2.1 Removing Degeneracy of Type *vertex-vertex* (Infrastructure)

Geometric Analysis For every vertex v in Q_{r-1}, we would like to have $d(v, v_r) > \lambda$. Therefore

$$F_{loc-v-v} = Set(vertex,\, Q_{r-1}) \oplus B(0, \lambda)$$

Volume Computation We are looking for the product of:

- The maximum number of vertices that a vertex might overlap within the same polyhedral surface, K.
- The volume of a λ-*expanded* vertex.

$$Volume(F_{loc-v-v}) \quad \leq \quad \frac{4}{3}\pi K \lambda^3$$

B.2.2 Removing Degeneracy of Type *vertex-edge* (Infrastructure)

Geometric Analysis Such a degeneracy can be induced in two ways:

1. The vertex is in Q_{r-1} while the edge is created by v_r.
2. The vertex is v_r while the edge is in Q_{r-1}.

The analysis is as follows:

1. For every vertex v in Q_{r-1}, and for every edge e that v_r creates, we would like to have $d(v, e) > \rho$.

   Let v_0 be one vertex of Q_{r-1}, and let v_1 be a vertex with which v_r is going to have a mutual edge. See Figure B.4.

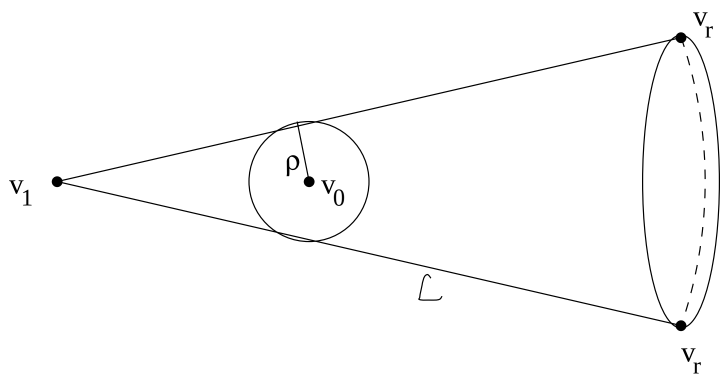


Figure B.4: Forbidden volume for the location of vertex V_r.

The edge (v_1, v_r) induces a degeneracy if it intersects $B(v_0, \rho)$. Therefore forbidden placements of v_r are in the rays starting in v_1 and intersecting $B(v_0, \rho)$. Those rays length is bounded by $\mathcal{L}$. The geometric shape of forbidden placements is a cone-like shape, with a spherical base.

Let $Cone(v_0, v_1, v_r)$ denote the cone of forbidden loci for v_r , where v_0, v_1, v_r are as defined above. Then

$$F_{loc\text{--}v\text{--}e\text{--}1} = \{Cone(v_0, v_1, v_r) | v_0, v_1 \in Set(\,vertex,\, Q_{r-1})\}$$

2. For every edge e in Q_{r-1}, we would like to have $d(e, v_r) > \rho$. Therefore

$$F_{loc\text{--}v\text{--}e\text{--}2} = Set(\,edge,\, Q_{r-1}) \oplus B(0, \rho)$$

Volume Computation We consider all ways for inducing such a degeneracy, according to the geometric analysis.

1. We bound the shape in Figure B.4 in a cone, as in Figure B.5.

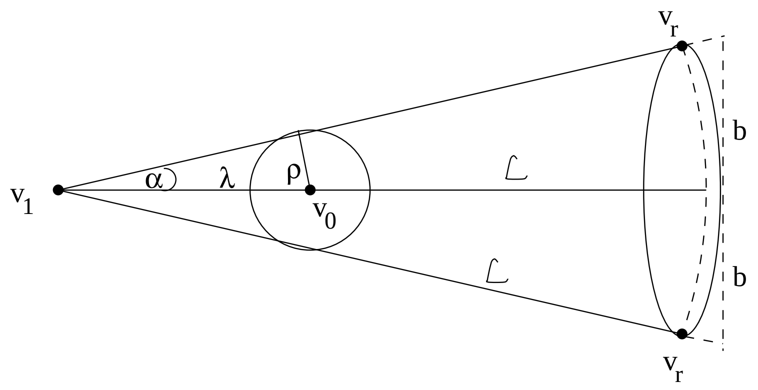


Figure B.5: Bounding the forbidden volume in a cone.

Let b be the radius at the base of the bounding cone, and let α be the angle at the apex of the cone as in Figure B.5. We are looking for a worst-case volume, which is obtained by the largest possible b. We obtain a bound for b by looking at the center cross section of the bounding cone. This cross section looks like a sector and has the same characteristics as the sector in case 1 of Section B.1.4.2, where the only difference is that ρ should be replaced by λ and ε should be replaced by ρ. Therefore we can use the bounds obtained for b in Section B.1.4.2:

$$b^2 \leq \frac{\rho^2}{\lambda^2 - \rho^2}\mathcal{L}^2$$

The volume of such a cone is $Volume(\,Cone(v_0, v_1, v_r)\,) = \frac{1}{3}\pi\frac{\rho^2}{\lambda^2-\rho^2}\mathcal{L}^3$.

Now we are looking for the product of:

- The maximum number of edges in a polyhedral surface, $3V - 3$.
- The maximum number of vertices that an edge might intersect, $2K$.
- $Volume(\,Cone(v_0, v_1, v_r)\,)$.

$$Volume(F_{loc\text{--}v\text{--}e\text{--}1}) \leq 2\pi K V \frac{\rho^2}{\lambda^2 - \rho^2}\mathcal{L}^3$$

2. We are looking for the product of:

- The maximum number of edges that a vertex might overlap within the same polyhedral surface, $2K$.
- The volume of a *ρ-expanded edge*.

$$Volume(F_{loc\text{--}v\text{--}e\text{--}2}) \quad \leq \quad 2\pi K \rho^2 (\mathcal{L} + \frac{4}{3}\rho)$$

B.2.3 Removing Degeneracy of Type *vertex-facet*

Geometric Analysis Such a degeneracy can be induced in two ways:

1. The vertex is in Q_{r-1} while the facet is created by v_r.
2. The vertex is v_r while the facet is in Q_{r-1}.

The analysis is as follows:

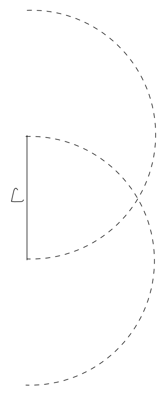


Figure B.6: Intersection of two half circles.

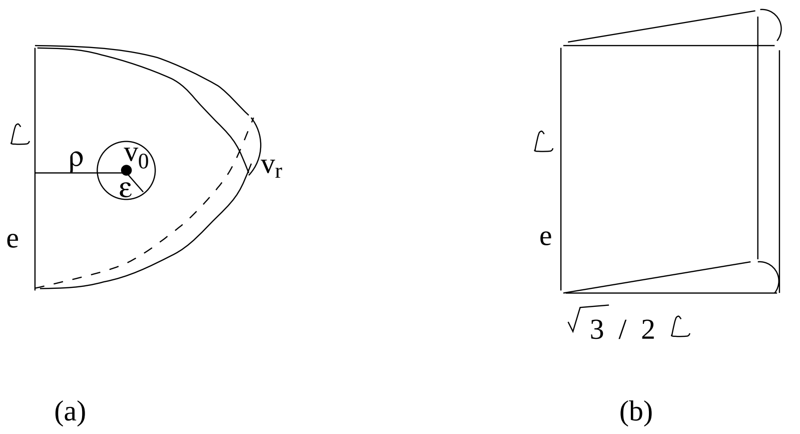


Figure B.7: (a) Forbidden loci for v_r. (b) Bounding the forbidden loci in a cylinder slice.

1. For every vertex v in Q_{r-1}, and for every facet f that v_r creates, we would like to have $d(v, f) > \varepsilon$.

   Let v_0 be one vertex of Q_{r-1}, and let e be the edge with which v_r shares a facet. In Figure B.6 we depict two half circles of radius $\mathcal{L}$. The lower and upper half circles (including their interiors) are the collection of edge endpoints for an edge starting from the lower and upper vertices of e respectively. The intersection of the half circles is the collection of all possible locations for v_r, i.e., all the places for a vertex connecting two edges that start in the endpoints of e. The distance from the furthermost point of the intersection to e is $\frac{\sqrt{3}}{2}\mathcal{L}$. The intersection is shown in Figure B.7(a), where we can see the three-dimensional shape that bounds the forbidden loci of v_r: Locating v_r inside the volume in Figure B.7(a) creates a facet which is too close to v_0 if it intersects $v_0 \oplus B(0, \varepsilon)$. Hence the forbidden loci are inside a slice where the planar facets are tangent to $v_0 \oplus B(0, \varepsilon)$. To ease our computations, we bound the volume of Figure B.7(a) in a cylinder slice shown in Figure B.7(b). Let $CSlice(v_0, e, v_r)$ denote the forbidden loci for v_r referring to edge e and vertex v_0.

   Hence
   $$F_{loc\text{-}v\text{-}f\text{-}1} = \{CSlice(v, e, v_r) \mid v \in Set(vertex, Q_{r-1}),\ e \in Set(edge, Q_{r-1})\}$$

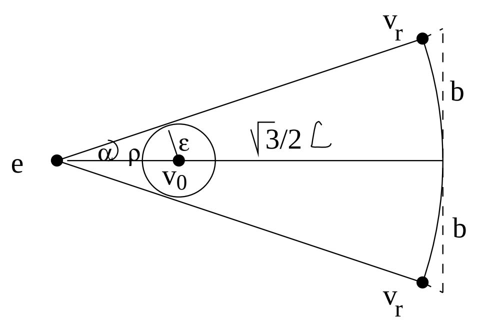


Figure B.8: A cross section of $CSlice(v_0, e, v_r)$.

where e is incident to v_r.

2. For every facet f in Q_{r-1}, we would like to have $d(f, v_r) > \varepsilon$. Therefore

$$F_{loc\text{--}v\text{--}f\text{--}2} = Set(\,facet,\, Q_{r-1}) \oplus B(0, \varepsilon)$$

Volume Computation We consider all ways for inducing such a degeneracy, according to the geometric analysis.

1. We denote the angle of the slice by 2α. The way of obtaining α is by analyzing the cross section of the cylinder slice, which is a disc sector. The sector itself is depicted in Figure B.8, and has the same characteristics as the sector in case 1 of Section B.1.4.2. (Notice that the sector radius in this section is not equal to the one in Section B.1.4.2, but it does not influence the ratio $\frac{2\alpha}{2\pi}$). Therefore we can use the bounds obtained for $\frac{2\alpha}{2\pi}$ in Section B.1.4.2:

$$\frac{2\alpha}{2\pi} \leq \frac{1}{\pi}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}$$

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

$$Volume(\,CSlice(e, v, vr_r)\,) = \frac{2\alpha}{2\pi} \cdot \mathcal{L}\pi(\frac{\sqrt{3}}{2}\mathcal{L})^2 \leq \frac{3}{4}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\mathcal{L}^3$$

   Now we are looking for the product of:

   - The maximum number of facets in a polyhedral surface, $2V - 2$.
   - The maximum number of vertices that a facet might intersect, $3K$.
   - $Volume(\,CSlice(e, v, v_r)\,)$

   Hence the volume of forbidden loci for v_r, referring to degeneracy of type *vertex-facet*, is:

$$volume(\,F_{loc\text{--}v\text{--}f\text{--}1}) \leq 4.5\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}KV\mathcal{L}^3$$

2. We are looking for the product of:

   - The maximum number of facets that a vertex might overlap within the same polyhedral surface, K.
   - The volume of an ε-*expanded* triangle.

$$Volume(F_{loc\text{--}v\text{--}f\text{--}2}) \quad \leq \varepsilon K(\mathcal{L}^2 + \tfrac{3}{2}\pi\mathcal{L}\varepsilon + \tfrac{10}{3}\pi\varepsilon^2)$$

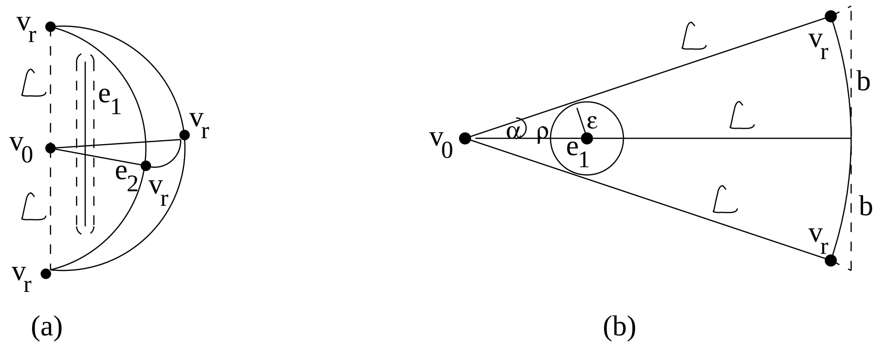


Figure B.9: (a) Forbidden loci for v_r. (b) Cross section of (a).

B.2.4 Removing Degeneracy of Type *edge-edge*

Geometric Analysis 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 $v_0 \in Set(vertex,\, Q_{r-1})$ denote the other end of e_2. The full range for creating an edge $e_2 = (v_0, v_r)$ is all the possible locations of v_r inside the sphere $B(v_0, \mathcal{L})$.

We aim to identify the sub-volume of this sphere, in which locating v_r induces a degeneracy of type *edge-edge*. In Figure B.9(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 sphere slice creates an edge e_2, which is 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 $e_1 \oplus B(0, \varepsilon)$.

Hence

$$F_{loc\text{–}e\text{–}e} = \{Slice(e, v, v_{r,}, \mathcal{L}) \mid e \in Set(edge,\, Q_{r-1}),\ v \in Set(vertex,\, Q_{r-1})\}$$

where *Slice* is defined in B.1.4.2 and v is incident to v_r.

Volume Computation The situation here is the same as in case 1 of Section B.1.4.2, and using the volume computed there, we get:

$$Volume(\,Slice(e, v, v_r, \mathcal{L})\,) \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\mathcal{L}^3$$

Now we are looking for the product of:

- The maximum number of edges in a polyhedral surface, $3V - 3$.
- The maximum number of edges that an edge might intersect, $2K$.
- $Volume(\,Slice(e, v, v_r, \mathcal{L})\,)$

Hence the volume of forbidden loci for v_r, referring to degeneracy of type *edge-edge*, is:

$$volume(F_{loc\text{–}e\text{–}e}) \leq 8\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}KV\mathcal{L}^3$$

B.3 Computing Forbidden Volumes for the Global Step

Suppose the incremental procedure of the global step has been carried out successfully for the first $j-1$ stages. The incremental procedure guarantees that there are no global inherent degeneracies in M_{j-1}. We would like to find a location for T_j so that no global degeneracy is induced.

Recall that $S_1 \oplus S_2$ denotes the Minkowski sum of sets S_1 and S_2, and that $S_1 \ominus S_2$ denotes the Minkowski sum of sets S_1 and $-S_2$.

B.3.1 Removing Degeneracy of Type *vertex-facet*

B.3.1.1 Volumes Induced by the Location of Two Polyhedral Terrains Relative to Each Other

Geometric Analysis Such a degeneracy can be induced in two ways:

1. The vertex is in M_{j-1} while the facet is in T_j.
2. The vertex is in T_j while the facet is in M_{j-1}.

The analysis is as follows:

1. For every vertex v in M_{j-1}, and for every facet f in T_j, we would like $d(v,f) > \varepsilon$. Therefore
$$F_{glob\text{-}v\text{-}f\text{-}1\text{-}1} = Set(vertex,\, M_{j-1}) \ominus Set(facet,\, T_j) \oplus B(0,\rho)$$
2. For every vertex v in T_j, and for every facet f in M_{j-1}, we would like $d(v,f) > \varepsilon$. Therefore
$$F_{glob\text{-}v\text{-}f\text{-}1\text{-}2} = Set(facet,\, M_{j-1}) \ominus Set(vertex\, T_j) \oplus B(0,\rho)$$

Volume Computation We consider all ways for inducing such a degeneracy, according to the geometric analysis.

1. We are looking for the product of:
   - The maximum number of facets in a polyhedral terrain, $2V-2$.
   - The maximum number of vertices that a facet might intersect, $3K$.
   - The volume of an *ε-expanded triangle.*

$$\begin{aligned} Volume(F_{glob\text{-}v\text{-}f\text{-}1\text{-}1}) &\leq (2V-2)\cdot 3K \cdot \varepsilon(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\varepsilon + \frac{10}{3}\pi\varepsilon^2) \\ &\leq 6VK\varepsilon(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\varepsilon + \frac{10}{3}\pi\varepsilon^2) \end{aligned}$$

2. We are looking for the product of:
   - The maximum number of vertices in a polyhedral terrain, V.
   - The maximum number of facets that a vertex might intersect, K.
   - The volume of an *ε-expanded triangle.*

$$Volume(F_{glob\text{-}v\text{-}f\text{-}1\text{-}2}) \leq KV\varepsilon(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\varepsilon + \frac{10}{3}\pi\varepsilon^2)$$

B.3.1.2 Volumes Induced by the Location of a Vertex Relative to a Boundary Edge (in Different Polyhedral Terrains) (Infrastructure)

Although this section deals with degeneracies of type *vertex-facet*, we deal here with degeneracies of type *vertex-edge.* The reason is that we see this section as dealing with facet boundaries, rather than edges.

Geometric Analysis Such a degeneracy can be induced in two ways:

1. The vertex is in M_{j-1} while the boundary edge is in T_j.
2. The vertex is in T_j while the boundary edge is in M_{j-1}.

The analysis is as follows:

1. For every vertex v in M_{j-1}, and for every boundary edge e in T_j, we would like $d(v, e) > \rho$. Therefore
$$F_{glob-v-f-2-1} = Set(\,vertex,\, M_{j-1}) \ominus Set(\,boundary\_edge,\, T_j) \oplus B(0, \rho)$$
2. For every vertex v in T_j, and for every boundary edge e in M_{j-1}, we would like $d(v, e) > \rho$. Therefore
$$F_{glob-v-f-2-2} = Set(\,boundary\_edge,\, M_{j-1}) \ominus Set(\,vertex,\, T_j) \oplus B(0, \rho)$$

Volume Computation We consider all ways for inducing such a degeneracy, according to the geometric analysis.

1. We are looking for the product of:
   - The maximum number of boundary edges in a polyhedral terrain, N.
   - The maximum number of vertices that an edge might intersect, $2K$.
   - The volume of a ρ-*expanded edge*.

$$\begin{aligned} Volume(F_{glob-v-f-2-1}) &\leq N \cdot 2K \cdot \pi\rho^2(\mathcal{L} + \frac{4}{3}\rho) \\ &\leq 2\pi N K \rho^2(\mathcal{L} + \frac{4}{3}\rho) \end{aligned}$$

2. We are looking for the product of:
   - The maximum number of vertices in a polyhedral terrain, V.
   - The maximum number of edges that a vertex might intersect, $2K$.
   - The volume of a ρ-*expanded edge*.

$$Volume(F_{glob-v-f-2-2}) \leq 2\pi V K \rho^2(\mathcal{L} + \frac{4}{3}\rho)$$

B.3.1.3 Volumes Induced by the Location of a Facet Relative to a Boundary Vertex (in Different Polyhedral Terrains) (Infrastructure)

Geometric Analysis Such a degeneracy can be induced in two ways:

1. The facet is in M_{j-1} while the boundary vertex is in T_j.
2. The facet is in T_j while the boundary vertex is in M_{j-1}.

The analysis is as follows:

1. For every facet f in M_{j-1}, and for every boundary vertex v in T_j, we would like $d(f, v) > \rho$. Therefore
$$F_{glob-v-f-3-1} = Set(\,facet,\, M_{j-1}) \ominus Set(\,boundary\_vertex,\, T_j) \oplus B(0, \rho)$$
2. For every facet f in T_j, and for every boundary vertex v in M_{j-1}, we would like $d(f, v) > \rho$. Therefore
$$F_{glob-v-f-3-2} = Set(\,boundary\_vertex,\, M_{j-1}) \ominus Set(\,facet,\, T_j) \oplus B(0, \rho)$$

Volume Computation We consider all ways for inducing such a degeneracy, according to the geometric analysis.

1. We are looking for the product of:
   - The maximum number of facets in a polyhedral terrain, $2V - 2$.
   - The maximum number of vertices that a facet might intersect, $3K$.
   - The volume of a ρ-*expanded triangle*.

$$\begin{aligned} Volume(F_{glob\text{-}v\text{-}f\text{-}3\text{-}1}) &\leq 2V \cdot 3K \cdot \rho(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\rho + \frac{10}{3}\pi\rho^2) \\ &\leq 6KV\rho(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\rho + \frac{10}{3}\pi\rho^2) \end{aligned}$$

2. We are looking for the product of:
   - The maximum number of boundary vertices in a polyhedral terrain, N.
   - The maximum number of facets that a vertex might intersect, K.
   - The volume of a ρ-*expanded triangle*.

$$\begin{aligned} Volume(F_{glob\text{-}v\text{-}f\text{-}3\text{-}2}) &\leq N \cdot K \cdot \rho(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\rho + \frac{10}{3}\pi\rho^2) \\ &\leq KN\rho(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\rho + \frac{10}{3}\pi\rho^2) \end{aligned}$$

B.3.1.4 Volumes Induced by the Location of a Vertex in M_{j-1} Relative to a Facet in $Con(T_j)$

Geometric Analysis For every vertex w in M_{j-1}, and for every connector H created by a perturbation of T_j, we would like to have $d(w, H) > \varepsilon$.

Let e, e', v, v', H be as defined in Section B.1.2.1. Let w be a vertex in M_{j-1}. A degeneracy occurs whenever H intersects the ball $B(w, \varepsilon)$. In Figure B.10 we can see the collection of all connectors H which are created by e and e' and intersect the ball $B(w, \varepsilon)$. Notice that the location of v and w is static, while the location of v' has not yet been determined. It is immediate that the location of the vertex v' can define the characteristics of the whole connector (i.e., its orientation, the location of u', etc.).

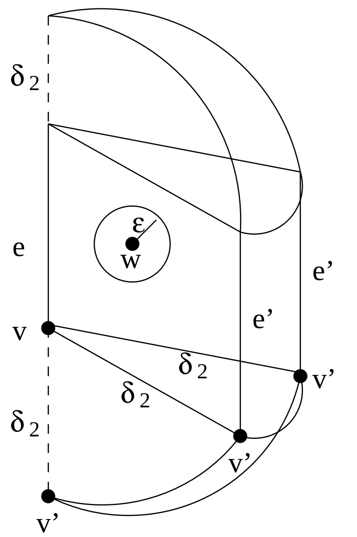


Figure B.10: Volume of forbidden connectors, where $d(v, v') < \delta_2$. (ε and δ_2 are magnified for clarity).

The range in which v' moves in order to create connectors such as in Figure B.10 is a slice of a sphere, whose axis is defined to be the line containing e. See Figure B.11. Locating v' inside the sphere slice creates a facet which is too close to w if it intersects $w \oplus B(0, \varepsilon)$. Hence the forbidden loci are inside a slice where the planar facets are tangent to $w \oplus B(0, \varepsilon)$. The sphere slice $Slice(w, v, v', \delta_2)$ denotes the forbidden placements for v', in the context of (almost) intersecting w, where $Slice$ is defined in B.1.4.2.

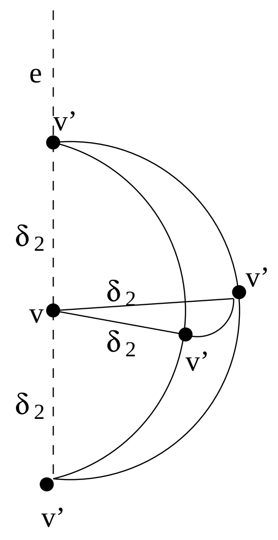


Figure B.11: $Slice(w, v, v', \delta_2)$.

Therefore

$$F_{glob\text{–}v\text{–}f\text{–}4} = \{Slice(w, v, v', \delta_2) \ominus v' \mid w \in Set(\,vertex,\ M_{j-1}), v' \in Set(\,boundary\_vertex,\ T_j)\}$$

(v is determined by v').

Volume Computation The sector fits case 1 of Section B.1.4.2:

- H passes through the straight line between v and v', and we would like this line not to intersect $B(w, \varepsilon)$.
- w's location is fixed.
- $d(v, w) > \rho$.

Figure B.12 is given here as a reminder. (The circle in the drawing is the two dimensional cross section of $B(w, \varepsilon)$).

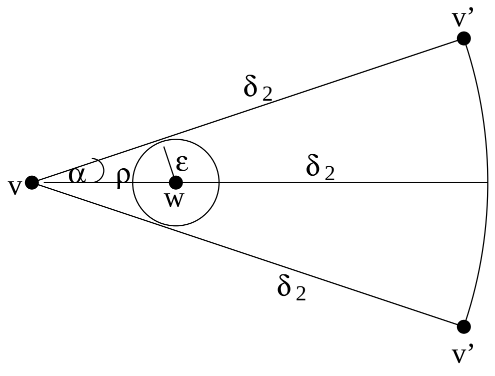


Figure B.12: A center cross section of $Slice(w, v, v', \delta_2)$.

Using the volume computed in B.1.4.2, we get:

$$Volume(\,Slice(w,v,v',\delta_2)\,) \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$$

Now we are looking for the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.
- The maximum number of vertices that a facet might intersect, $3K$.
- $Volume(\,Slice(w,v,v',\delta_2)\,)$.

$$Volume(F_{glob\text{–}v\text{–}f\text{–}4}) \leq 4KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$$

B.3.1.5 Volumes Induced by the Location of a Vertex in T_j Relative to a Facet in $Con(T_j)$

Geometric Analysis For every vertex w in T_j, and for every connector H created by a perturbation of T_j, we would like to have $d(w,H) > \varepsilon$.

Let e,e',v,v',H be as defined in Section B.1.2.1. Let w be a vertex in T_j. A degeneracy occurs whenever H intersects the ball $B(w,\varepsilon)$. See Figure B.13. Notice that the location of v is static, while the location of v' has not yet been determined. Unlike in Section B.3.1.4, the location of w has not yet been determined and depends on the location of v'.

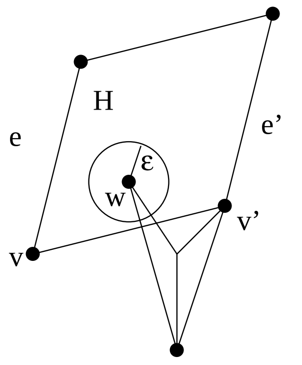


Figure B.13: A degeneracy induced by a vertex in T_j which is too close to a connector of T_j.

Let us look at the sphere slice in Figure B.11. We choose the axis of the sphere slice to be the line containing e. Locating v' inside the sphere slice creates a facet which is too close to w if it intersects $w \oplus B(0,\varepsilon)$. Hence the forbidden loci are inside a slice where the planar facets are tangent to $w \oplus B(0,\varepsilon)$. The sphere slice $Slice(w,v,v',\delta_2)$ defines the forbidden placements for v', in the context of (almost) intersecting w, where $Slice$ is defined in B.1.4.2.

We first analyze the center cross section of the sphere slice.

Let $\vec{x}$ be the vector $v' - w$.

Simple case: $\vec{x}$ is perpendicular to e.
Due to the fact that $\vec{x}$ is perpendicular to e, $\vec{x}$ is contained in the plane of the center cross section.
The sector fits case 2 of Section B.1.4.2:

- H passes through the straight line between v and v', and we would like this line not to intersect $B(w,\varepsilon)$.
- w is rigidly attached to v'.
- $d(v',w) > \rho$, only for the simple case (for now).

Figure B.14 is given here as a reminder.

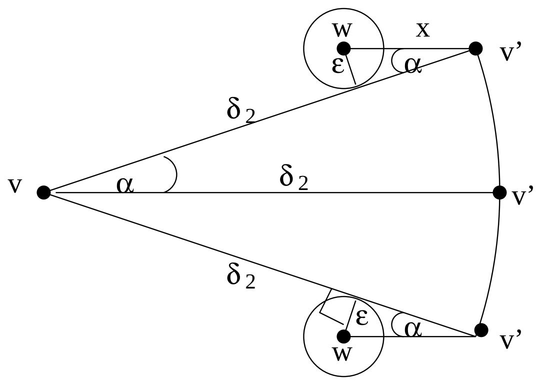


Figure B.14: A cross section of the forbidden volume.

General case: $\vec{x}$ is not necessarily perpendicular to e.

Let us create a line passing through w and parallel to e. Let w' be the intersection point of this line with a plane passing through v' and perpendicular to e. See Figure B.15. w' is at the same distance from the connector H as w is, so it induces the same forbidden loci for v' as w does. Thus we have reduced the general case to the simple case with w' now playing the role of w in the simple case.

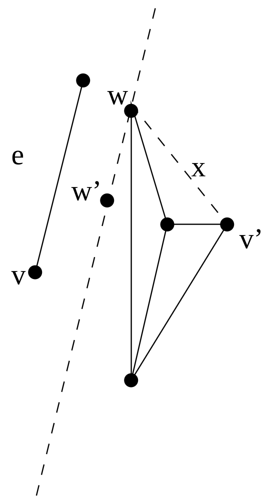


Figure B.15: Obtaining the simple case out of the general case.

Notice that also in the general case we have $d(v', w') > \rho$: We have $d(e', w) > \rho$ and since the line (w, w') is parallel to e', we have $d(e', w') = d(e', w) > \rho$. v' is on e' so we have $d(v', w') > \rho$. Therefore case 2 of Section B.1.4.2 still holds.

Therefore

$$F_{glob-v-f-5} = \{Slice(w, v, v', \delta_2) \ominus v' \mid w \in Set(vertex, T_j),\ v' \in Set(boundary\_vertex, T_j)\}$$

(v is determined by v').

Volume Computation Using the volume computed in B.1.4.2, we get:

$$Volume(Slice(w, v, v', \delta_2)) \leq \frac{4}{3} \frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}} \delta_2^3$$

Now we are looking for the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.
- The maximum number of vertices that a facet might intersect, $3K$.
- $Volume(\,Slice(w, v, v', \delta_2)\,)$.

$$Volume(F_{glob\text{--}v\text{--}f\text{--}5}) \leq 4KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$$

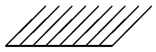
B.3.2 Removing Degeneracy of Type *edge-edge*

B.3.2.1 Volumes Induced by the Location of Two Polyhedral Terrains Relative to Each Other

Geometric Analysis For every edge e_1 in M_{j-1}, and for every edge e_2 in T_j, we would like to have $d(e_1, e_2) > \varepsilon$. Therefore

$$F_{glob\text{--}e\text{--}e\text{--}1} = Set(\,edge,\, M_{j-1}) \ominus Set(\,edge, T_j) \oplus B(0, \varepsilon)$$

Volume Computation The Minkowski sum of 2 edges, illustrated in the following drawing, is a parallelogram:



We are looking for the product of:

- The maximum number of edges in a polyhedral terrain, $3V - 3$.
- The maximum number of edges that an edge might intersect, $2K$.
- The volume of an ε-*expanded parallelogram.*

$$\begin{aligned} Volume(F_{glob\text{--}e\text{--}e\text{--}1}) &\leq (3V-3)2K \cdot 2\varepsilon(\mathcal{L}^2 + \mathcal{L}\pi\varepsilon + 2\pi\varepsilon^2) \\ &\leq 12KV\varepsilon(\mathcal{L}^2 + \pi\mathcal{L}\varepsilon + 2\pi\varepsilon^2) \end{aligned}$$

B.3.2.2 Volumes Induced by the Location of an Edge in M_{j-1} Relative to an Edge in $Con(T_j)$

Geometric Analysis For every edge e_1 in M_{j-1}, and for every edge e_2 in a connector H created by a perturbation of T_j, we would like to have $d(e_1, e_2) > \varepsilon$.

Let e, e', v, v', u, u', H be as defined in Section B.1.2.1. Two edges that end in v' are internal to H: (v, v') and (u, v'). Figure B.16 is given as a reminder.

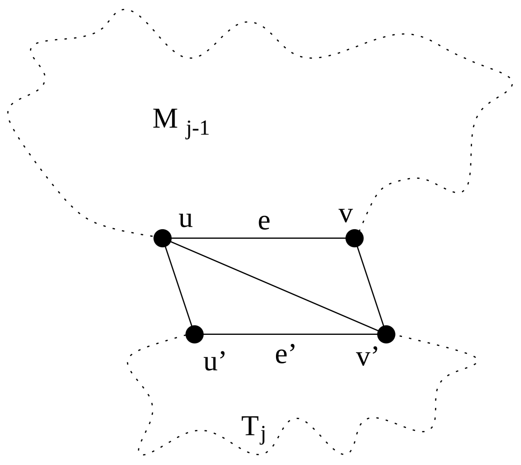


Figure B.16: A typical connector.

Let e_1 be an edge in M_{j-1}. A degeneracy occurs whenever (v, v') or (u, v') intersects the ε-*expanded edge* around e_1. So e_2 mentioned above can be either (v, v') or (v, u'). We do not consider (u, u') right now since we try to bound the forbidden volume for v'; u' will be taken into consideration in the final product.

Although the diagonal of H could be (v,u') instead of (u,v'), we don't care since the volume of the forbidden placement would be the same for symmetry reasons.

Notice that the locations of v and e_1 are static, while the location of v' has not yet been determined.

Let us look at the sphere slice in Figure B.17. We choose the axis of the sphere slice to be a line parallel to e_1 passing through v. Locating v' inside the sphere slice creates an edge, which is 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 $e_1 \oplus B(0,\varepsilon)$. The sphere slice $Slice(e_1,x,v',len)$ denotes the volume of forbidden placements for v', in the context of (almost) intersecting e_1, where $Slice$ is defined in Section B.1.4.2, x is either v or u, and len is either δ_2 or $\mathcal{L}+\delta_2$ respectively.

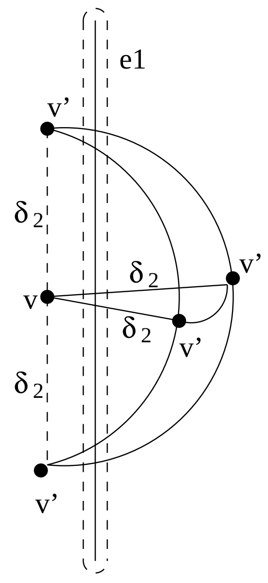


Figure B.17: A degeneracy occurs as long as the the ε-*expanded edge* around e_1 is inside the sphere slice.

Therefore

$$F_{glob\text{-}e\text{-}e\text{-}2} = \{\, ((Slice(e_1,v,v',\delta_2) \cup Slice(e_1,u,v',\mathcal{L}+\delta_2)) \ominus v') \mid v' \in Set(\mathit{boundary\_vertex},\, T_j),\ e_1 \in Set(\mathit{edge},\, M_{j-1})\}$$

(v and u are determined by v').

Volume Computation The sector fits case 1 of Section B.1.4.2:

- We would like (v,v') not to intersect the ε-*expanded edge* around e_1.
- e_1's location is fixed.
- $d(v,e_1) > \rho$.

Figure B.18 is given here as a reminder. Notice that the little circle is a horizontal cross section of the ε-*expanded edge* around e_1.

Using the volume computed in Section B.1.4.2, we get:

$$Volume(Slice(e_1,v,v',\delta_2)) \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$$

$$Volume(Slice(e_1,u,v',\mathcal{L}+\delta_2)) \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\mathcal{L}+\delta_2)^3$$

Now we are looking for the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.

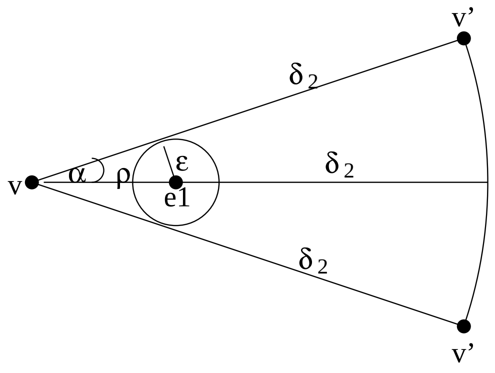


Figure B.18: Center cross section of forbidden volume.

- The maximum number of edges that an edge might intersect, $2K$.
- $Volume(\,Slice(e_1, v, v', \delta_2) \cup Slice(e_1, u, v', \mathcal{L} + \delta_2)\,)$.

$$\begin{aligned} Volume(F_{glob\text{–}e\text{–}e\text{–}2}) &\leq N \cdot 2K \cdot (\frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3 + \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\mathcal{L}+\delta_2)^3) \\ &\leq \frac{8}{3}KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\delta_2^3 + (\mathcal{L}+\delta_2)^3) \end{aligned}$$

B.3.2.3 Volumes Induced by the Location of an Edge in T_j Relative to an Edge in $Con(T_j)$

Geometric Analysis For every edge e_1 in T_j, and for every edge e_2 in a connector H created by a perturbation of T_j, we would like to have $d(e_1, e_2) > \varepsilon$.

Let e, e', v, v', u, u', H be as defined in Section B.1.2.1. As explained in the previous section, we take care of the edges which are internal to H ((v, v') and (u, v')) and ignore (v, u').

Let e_1 be an edge in T_j. A degeneracy occurs whenever (v, v') or (u, v') intersects the ε-*expanded edge* around e_1. So e_2 mentioned above can be either (v, v') or (v, u'). See Figure B.19.

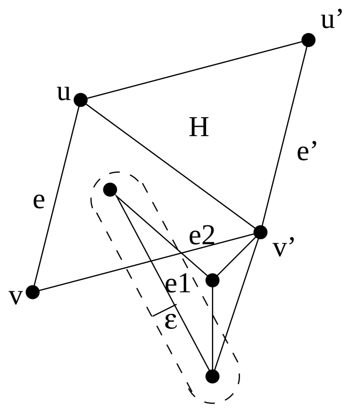


Figure B.19: A degeneracy induced by an edge in T_j which is too close to an edge in $Con(T_j)$.

Notice that the location of v is static, while the location of v' has not yet been determined. Unlike in Section B.3.2.2, the location of e_1 has not yet been determined and depends on the location of v'.

Let us look at the sphere slice in Figure B.20. We choose the axis of the sphere slice to be a line parallel to e_1. Locating v' inside the sphere slice creates an edge, which is 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 $e_1 \oplus B(0, \varepsilon)$.

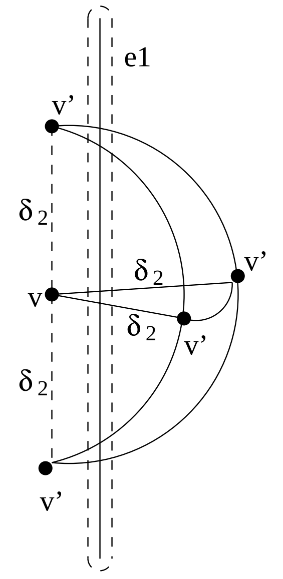


Figure B.20: A degeneracy occurs unless the ε-*expanded edge* around e_1 is outside the sphere slice.

Let us look at the center cross section of the sphere slice.
The sector fits case 2 of Section B.1.4.2:

- We would like (v, v') not to intersect the ε-*expanded edge* around e_1.
- e_1 is rigidly attached to v'.
- $d(v', e_1) > \rho$.

Figure B.21 is given here as a reminder. Notice that the little circle is a horizontal cross section of the ε-*expanded edge* around e_1.

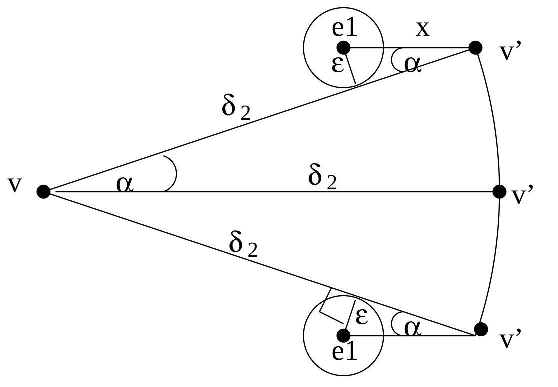


Figure B.21: Center cross section of forbidden volume.

The same analysis holds for the edge (u, v'). The only difference is that for (u, v') the radius of the sphere is $\mathcal{L} + \delta_2$.

The sphere slice $Slice(e_1, x, v', len)$ denotes the volume of forbidden placements for v', in the context of (almost) intersecting e_1, where $Slice$ is defined in Section B.1.4.2, x is either v or u, and len is either δ_2 or $\mathcal{L} + \delta_2$ respectively.

Therefore

$$F_{glob\text{–}e\text{–}e\text{–}3} = \{\, ((Slice(e_1, v, v', \delta_2) \cup Slice(e_1, u, v', \mathcal{L}+\delta_2)) \ominus v') \mid v' \in Set(\,boundary\_vertex,\, T_j),\ e_1 \in Set(\,edge,\, T_{j-1})\}$$

(v and u are determined by v').

Volume Computation Using the volume computed in Section B.1.4.2, we get:

$$Volume(Slice(e_1, v, v', \delta_2)) \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\delta_2^3$$

$$Volume(Slice(e_1, u, v', \mathcal{L} + \delta_2)) \leq \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}(\mathcal{L} + \delta_2)^3$$

Now we are looking for the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.
- The maximum number of edges that an edge might intersect, $2K$.
- $Volume(\,Slice(e_1, v, v', \delta_2) \cup Slice(e_1, u, v', \mathcal{L} + \delta_2)\,)$.

$$\begin{aligned} Volume(F_{glob-e-e-3}) &\leq N \cdot 2K \cdot (\frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\delta_2^3 + \frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}(\mathcal{L} + \delta_2)^3) \\ &\leq \frac{8}{3}KN\frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}(\delta_2^3 + (\mathcal{L} + \delta_2)^3) \end{aligned}$$

B.3.3 Removing Degeneracy of Type *edge-segment*

B.3.3.1 Volumes Induced by the Location of Two Polyhedral Terrains Relative to Each Other

Geometric Analysis Such a degeneracy can be induced in two ways:

1. The edge is in T_j, while the segment is in M_{j-1}.
2. The edge is in M_{j-1}. The segment is created by an intersection of one facet from M_{j-1} and one facet from T_j.

The analysis is as follows:

1. For every edge e in T_j, and for every segment s in M_{j-1}, we would like to have $d(e, s) > \varepsilon$. Therefore

$$F_{glob-e-s-1-1} = Set(\,segment,\, M_{j-1}) \ominus Set(\,edge,\, T_j) \oplus B(0, \varepsilon)$$

2. For every intr2 g in M_{j-1}, and for every facet f in T_j, we would like to have $d(f, g) > \rho$. (The analysis is easier when seperating factors from M_{j-1} and T_j. In this case, the factors from M_{j-1} are a facet and an edge, and their intersection is an intr2). Therefore

$$F_{glob-e-s-1-2} = Set(\,intr2,\, M_{j-1}) \ominus Set(\,facet,\, T_j) \oplus B(0, \rho)$$

Volume Computation We consider all ways for inducing such a degeneracy, according to the geometric analysis.

1. The Minkowski sum of 2 edges is a parallelogram.

   We are looking for the product of:

   - The maximum number of edges in a polyhedral terrain, $3V - 3$.
   - The maximum number of segments that an edge might intersect, $\binom{K}{2}$.

- The volume of an ε-*expanded parallelogram.*

$$\begin{aligned} Volume(F_{glob\text{–}e\text{–}s\text{–}1\text{–}1}) &\leq (3V-3)\binom{K}{2}2\varepsilon(\mathcal{L}^2+\pi\mathcal{L}\varepsilon+2\pi\varepsilon^2) \\ &\leq 3VK^2\varepsilon(\mathcal{L}^2+\pi\mathcal{L}\varepsilon+2\pi\varepsilon^2) \end{aligned}$$

2. We are looking for the product of:

   - The maximum number of facets in a polyhedral terrain, $2V-2$.
   - The maximum number of intr2 that a facet might intersect, $2\binom{K}{2}$.
   - The volume of an ε-*expanded triangle.*

$$\begin{aligned} Volume(F_{glob\text{–}e\text{–}s\text{–}1\text{–}2}) &\leq (2V-2)\cdot 2\binom{K}{2}\cdot\varepsilon(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\varepsilon+\frac{10}{3}\pi\varepsilon^2) \\ &\leq 2VK^2\varepsilon(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\varepsilon+\frac{10}{3}\pi\varepsilon^2) \end{aligned}$$

B.3.3.2 Volumes Induced by the Location of a Segment in M_{j-1} Relative to an Edge in $Con(T_j)$

For every segment s in M_{j-1}, and for every edge e_2 in a connector H created by a perturbation of T_j, we would like to have $d(s,e_2)>\varepsilon$.

The analysis and computation are exactly as in B.3.2.2, when replacing e_1 by s.

Therefore the volume of $F_{glob\text{–}e\text{–}s\text{–}2}$ is the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.
- The maximum number of segments that an edge might intersect, $\binom{K}{2}$.
- $Volume(\,Slice(s,v,v',\delta_2)\cup Slice(s,u,v',\mathcal{L}+\delta_2)\,)$.

$$\begin{aligned} Volume(F_{glob\text{–}e\text{–}s\text{–}2}) &\leq N\cdot\binom{K}{2}\cdot(\frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3+\frac{4}{3}\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\mathcal{L}+\delta_2)^3) \\ &\leq \frac{2}{3}K^2N\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\delta_2^3+(\mathcal{L}+\delta_2)^3) \end{aligned}$$

B.3.3.3 Volumes Induced by the Location of an Edge in M_{j-1} Relative to a Segment Created by an Intersection of One Facet from M_{j-1} and One Facet From $Con(T_j)$

For every intr2 g in M_{j-1}, and for every connector H created by a perturbation of T_j, we would like to have $d(g,H)>\varepsilon$. (We are seperating the factors from M_{j-1} and from $Con(T_j)$, and the intr2 is obtained from the factors in M_{j-1}).

The analysis and computation are exactly as in B.3.1.4, when replacing w by g.

Therefore the volume of $F_{glob\text{–}e\text{–}s\text{–}3}$ is the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.
- The maximum number of intr2 that a facet might intersect, $2\binom{K}{2}$
- $Volume(\,Slice(g,v,v',\delta_2)\,)$.

$$Volume(F_{glob\text{–}e\text{–}s\text{–}3})\leq\frac{4}{3}K^2N\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$$

B.3.3.4 Volumes Induced by the Location of an Edge in T_j Relative to a Segment Created by an Intersection of One Facet from M_{j-1} and One Facet from $Con(T_j)$

Geometric Analysis Let S_4 be the set of intersection segments which are created by a facet in M_{j-1} and a facet in $Con(T_j)$.

For every segment s in S_4, and for every edge E in T_j, we would like to have $d(s, E) > \varepsilon$.

Let e, e', v, v', H be as defined in Section B.1.2.1. Let f be a facet in M_{j-1}. Let E be an edge in T_j. A degeneracy occurs whenever the intersection of H and f intersects the ε-*expanded edge* around E. See Figure B.22.

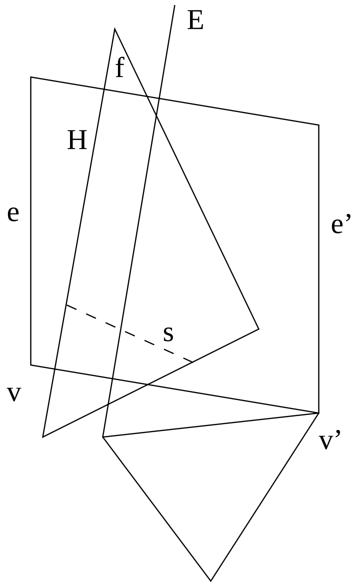


Figure B.22: The location of an edge E in T_j relative to a segment s created by a facet f in M_{j-1} and a facet H in $Con(T_j)$.

Fixed elements:

- The edge e
- The facet f
- The relative distance and direction from v' to E (i.e. the location of both v' and E has not yet been determined, but the location of each one relative to the other is fixed: v' and E are both in T_j, which is perturbed as a rigid object).

For convenience, we transform the coordinate system so that e is in the x axis, and E is in a plane parallel to the xy plane. See Figure B.23.

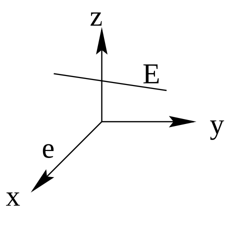


Figure B.23: A transformation of the coordinate system placing e and E in convenient positions.

Let us look at the plane PL_θ that passes through e and has an angle of θ relative with the xy plane. Figure B.24 shows a cross section of the yz plane showing PL_θ.

Let s_θ be the intersection segment created by PL_θ and f.

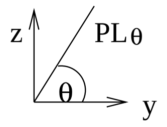


Figure B.24: A cross section of the yz plane showing PL_θ.

We start by looking at connectors H which are in the plane PL_θ. v' creates such connectors if it moves only in two dimensions, inside PL_θ.

The movement of v' moves E as well. Let i_θ be a point on E, with coordinates defined as an offset from v'. i_θ is defined to be the intersection point of E and PL_θ. i_θ remains fixed (relative to v') as long as v' moves in the plane PL_θ.

We move v' so that E intersects s_θ. The geometric location of v' that induces an intersection of E and s_θ is the segment $s_\theta \ominus \{i_\theta\}$. See Figure B.25.

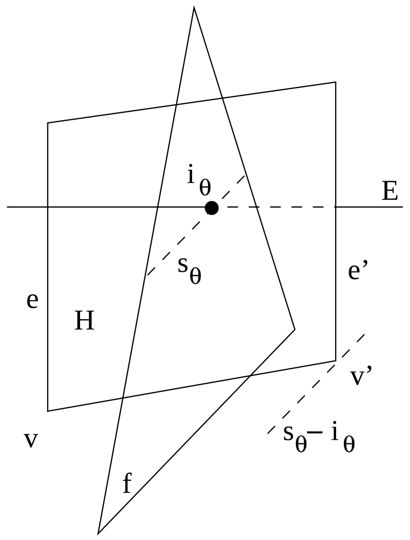


Figure B.25: v' induces an intersection of E and s_θ if it moves along the segment $s_\theta \ominus \{i_\theta\}$.

Next we refer to the fact that our interest is not only in E, but in the ε-*expanded edge* around E. An intersection of an ε-*expanded edge* with a plane is an intersection of a cylinder with a plane, which produces an ellipse.

We change here the notation i_θ: It does not denote a point anymore; it denotes a cylinder cross section. The geometric location of v' that induces an intersection of $Exp(\varepsilon, E)$ and s_θ is therefore the planar Minkowski sum $s_\theta \ominus i_\theta$. See Figure B.26.

Let the possible range for θ be $[\alpha, \beta]$. (α and β are computed later).

For each $\theta \in [\alpha, \beta]$ we get a different planar Minkowski difference $s_\theta \ominus i_\theta$. The range $[\alpha, \beta]$ is continuous, which implies that the areas $s_\theta \ominus i_\theta$ are continuous and their union forms a three-dimensional volume. See Figure B.27.

Let $Loci(E, f, v')$ denote the forbidden loci for v', as analyzed above, with E, f as defined above. Then

$$F_{glob\text{-}e\text{-}s\text{-}4} = \{Loci(E, f, v') \mid E \in Set(edge, T_j),\ f \in Set(facet, M_{j-1}),\ v' \in Set(boundary\_vertex, M_{j-1})\}$$

Volume Computation

Defining the Endpoints of *E* We use the terms 'leftmost' and 'rightmost' as seen when looking in the negative x direction. Let (x_0, y_0, z_0) be the offset of the leftmost vertex of E from v'. Let $(a, b, 0)$ be a unit direction vector of E. (The direction in the z axis is zero since E is in the xy plane). The worst case E is the longest E. Therefore we choose an E of length $\mathcal{L}$. See Figure B.28.

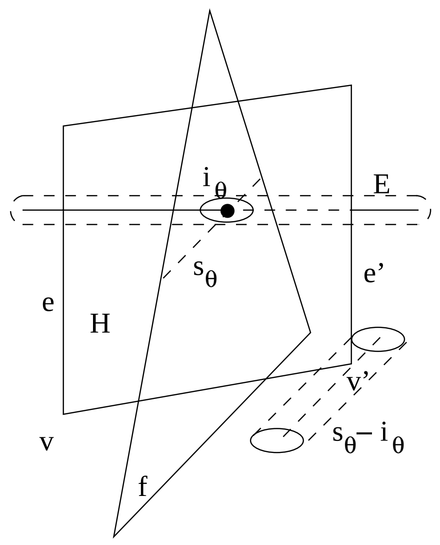


Figure B.26: v' induces an intersection of $Exp(\varepsilon, E)$ and s_θ if it moves along the planar Minkowski difference $s_\theta \ominus i_\theta$.

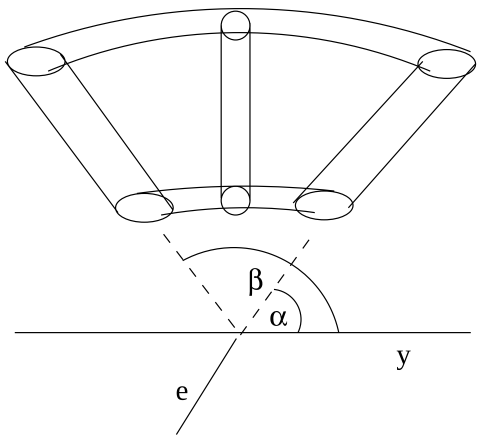


Figure B.27: The volume where v' induces intersections of $Exp(\varepsilon, E)$ and f. In this example f is perpendicular to e.

Computing the Range $[\alpha, \beta]$ Let us find the range $[\alpha, \beta]$ for connectors H. We are interested only in connectors that intersect the edge E.

The angle α is defined when the leftmost vertex of E touches the connector H (Figure B.29(a)) and the angle β is defined when the rightmost vertex of E touches the connector H (Figure B.29(b)). Recall that E is rigidly attached to v', and that (x_0, y_0, z_0) are defined relative to v' and not in absolute coordinates.

It is immediate that

$\tan\alpha = \frac{z_0}{y_0}$, $\sin\alpha = -\frac{z_0}{\sqrt{y_0^2+z_0^2}}$, $\cos\alpha = -\frac{y_0}{\sqrt{y_0^2+z_0^2}}$

$\tan\beta = \frac{z_0}{y_0+\mathcal{L}b}$, $\sin\beta = -\frac{z_0}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}}$, $\cos\beta = -\frac{y_0+\mathcal{L}b}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}}$

Considering the Location and Size of f We now know the range of the angle θ, but we do not know where in this range $Exp(\varepsilon, E)$ intersects f. Since we are looking for a worst case, we can always find such a facet f which intersects $Exp(\varepsilon, E)$ for every $\theta \in [\alpha, \beta]$. Therefore we assume that the range for θ is not smaller than $[\alpha, \beta]$.

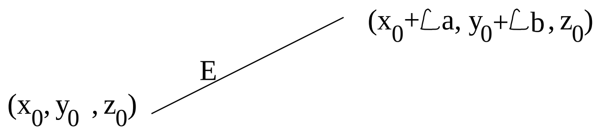


Figure B.28: Endpoints of E (as offset from v').

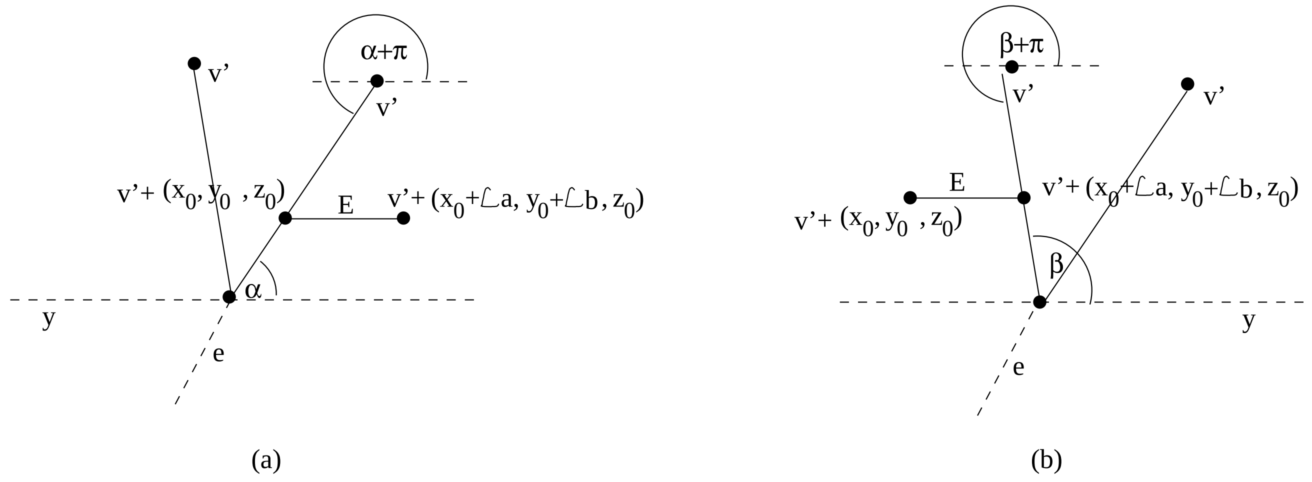


Figure B.29: A projection on the yz plane. (a) Defining the angle α, (b) defining the angle β.

Computing the Intersection Ellipse of $Exp(\varepsilon, E)$ and f It is immediate that the length of the minor axis of the ellipse i_θ, which is the intersection of $Exp(\varepsilon, E)$ and PL_θ, is 2ε.

Claim B.1 *For each angle $\theta \in [\alpha, \beta]$, the length of the major axis of the ellipse i_θ, is $\frac{2\varepsilon}{\sin\theta \cdot b}$*

Proof:
A unit vector which is perpendicular to PL_θ is $(0, -\sin\theta, \cos\theta)$. It is shown in Figure B.30 where we have
$x = 0$,
$y = \cos(\frac{\pi}{2} + \theta) = -\sin\theta$ and
$z = \sin(\frac{\pi}{2} + \theta) = \cos\theta$.

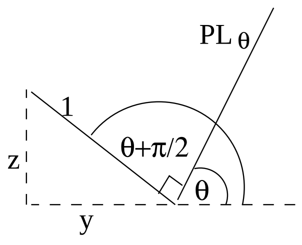


Figure B.30: A unit perpendicular to PL_θ, shown in a cross section of the yz plane.

For convenience, we use the opposite direction vector, $(0, \sin\theta, -\cos\theta)$.
Let ϕ be the angle between the perpendicular to PL_θ and E.
$\cos\phi = (0, \sin\theta, -\cos\theta) \cdot (a, b, 0) = 0 \cdot a + \sin\theta \cdot b + (-\cos\theta) \cdot 0 = \sin\theta \cdot b$
Let l be the length of the major axis of the intersection ellipse. Using the angles in Figure B.31(b) we get $\frac{2\varepsilon}{l} = \cos\phi$, and therefore

$$l = \frac{2\varepsilon}{\cos\phi} = \frac{2\varepsilon}{\sin\theta \cdot b}$$

■

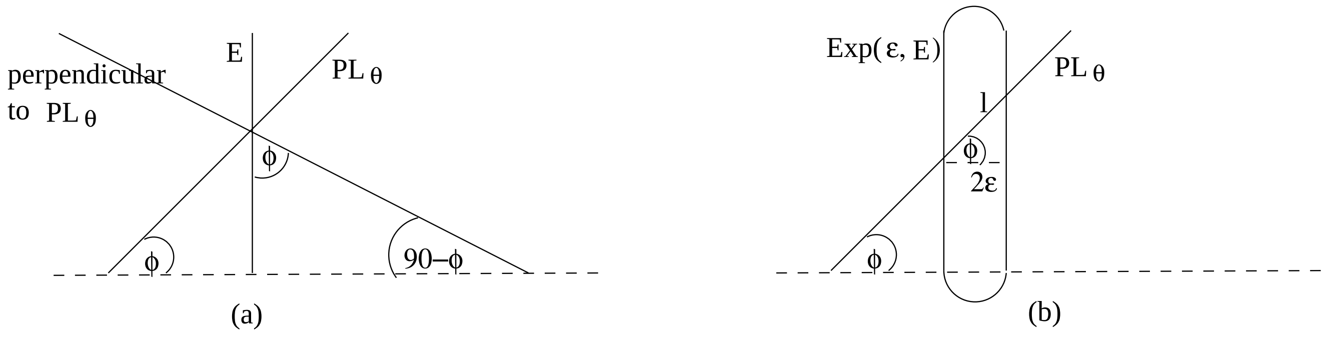


Figure B.31: A planar cross section created by E and the perpendicular to PL_θ. (a) Showing why the angle ϕ in (b) is correct, (b) calculating the length of the ellipse.

Discussing the Area of $s_\theta \ominus i_\theta$ The Minkowski difference is illustrated in Figure B.32. The area $s_\theta \ominus i_\theta$

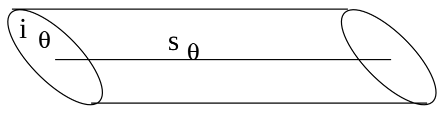


Figure B.32: The Minkowski difference $s_\theta \ominus i_\theta$.

can be bounded in a parallelogram. The length of the horizontal edge of the parallelogram ('horizontal' is used according to Figure B.32) is bounded by $length(s_\theta) + 2\varepsilon$, where 2ε bounds the length of the minor axis of the ellipse i_θ. The maximum area (since we are looking for a worst case) is obtained when the ellipse is vertical, so we get a maximum area of $\frac{2\varepsilon}{\sin\theta \cdot b} \cdot (length(s_\theta) + 2\varepsilon)$.

Bounding $length(s_\theta) + 2\varepsilon$ As a matter of fact, the length of s_θ is not interesting for us, because we are dealing with movements of v', which are bounded by the length $d(v, v')$. According to the inequality $\rho < d(v, v') < \delta_2$, we infer that the length of the movement of v' is bounded by $\delta_2 - \rho$. Since $\rho > 2\varepsilon$ (see Section B.4.1), we can use the bound δ_2.

Computing the Volume that Induces Degeneracies For each angle θ, i_θ is 'dragged' along the segment s_θ. When dealing with the continuous region $[\alpha, \beta]$, we need to compute an integral. Thinking of an integral as the sum of very tiny volumes, we can realize that each volume is a surface of revolution for the angle $d\theta$. The formula used inside the following integral is the formula for surface of revolution.

$$\begin{aligned} Volume(Loci(E, f, v')) &\leq \int_\alpha^\beta \frac{2\varepsilon}{\sin\theta \cdot b} \frac{\delta_2^2}{2} d\theta \\ &= \frac{\varepsilon\delta_2^2}{b} \ln(\csc\theta - \cot\theta) \mid_\alpha^\beta \\ &= \frac{\varepsilon\delta_2^2}{b} \ln \frac{\sqrt{(y_0 + \mathcal{L}b)^2 + z_0^2} + y_0 + \mathcal{L}b}{\sqrt{y_0^2 + z_0^2} + y_0} \end{aligned}$$

We next look for an upper bound for the volume.

$Volume(Loci(E, f, v')) = \frac{\varepsilon\delta_2^2}{b} \ln \frac{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}+y_0+\mathcal{L}b}{\sqrt{y_0^2+z_0^2}+y_0} = \frac{\varepsilon\delta_2^2}{b} \ln(1 + \frac{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-\sqrt{y_0^2+z_0^2}+\mathcal{L}b}{\sqrt{y_0^2+z_0^2}+y_0})$

Let us look at the triangle depicted in Figure B.33. According to triangle inequality, we get $\sqrt{(y_0 + \mathcal{L}b)^2 + z_0^2} - \sqrt{y_0^2 + z_0^2} \leq \mathcal{L}b$. Therefore $Volume(Loci(E, f, v')) \leq \frac{\mathcal{L}\varepsilon\delta_2^2}{\mathcal{L}b} \ln(1 + \frac{2}{\sqrt{y_0^2+z_0^2}+y_0}\mathcal{L}b)$.

$\frac{2}{\sqrt{y_0^2+z_0^2}+y_0} > 0$ and according to the inequality $\ln(1+x) \leq x$ (for any $x > 0$) we get that $Volume(Loci(E, f, v')) \leq \frac{\varepsilon\delta_2^2}{b} \cdot \frac{2}{\sqrt{y_0^2+z_0^2}+y_0}\mathcal{L}b = 2\mathcal{L}\delta_2^2 \frac{\varepsilon}{\sqrt{y_0^2+z_0^2}+y_0}$

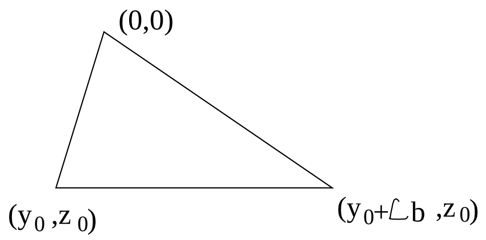


Figure B.33: Values of triangle vertices, used for triangle inequality.

$d(v', E) > \rho$ and therefore $\frac{1}{\sqrt{y_0^2+z_0^2}+y_0} < \frac{1}{\rho}$, so we get

$$Volume(Loci(E, f, v')) \leq 2\mathcal{L}\delta_2^2\frac{\varepsilon}{\rho}$$

We are looking for the product of:

- The maximum number of edges in a polyhedral terrain, $3V - 3$.
- The maximum number of segments that an edge might intersect, $\binom{K}{2}$.
- $Volume(\,Loci(E, f, v')\,)$

$$Volume(F_{glob\text{--}e\text{--}s\text{--}4}) \leq 3VK^2\delta_2^2\mathcal{L}\frac{\varepsilon}{\rho}$$

B.3.3.5 Volumes Induced by the Location of an Edge in M_{j-1} relative to a Segment Created by an Intersection of One Facet from T_j and One Facet from $Con(T_j)$

The geometric analysis and the volume computation in this section are similar to the ones in Section B.3.3.4, but not identical, and although it might seem as partially repeating Section B.3.3.4, there are significant differences which should be noticed.

Geometric Analysis Let S_5 be the set of intersection segments which are created by a facet in T_j and a facet in $Con(T_j)$.

For every segment s in S_5, and for every edge E in M_{j-1}, we would like to have $d(s, E) > \varepsilon$.

Let e, e', v, v', H be as defined in Section B.1.2.1. Let f be a facet in T_j. Let E be an edge in M_{j-1}. A degeneracy occurs whenever the intersection of H and f intersects the *ε-expanded edge* around E. See Figure B.34.

Fixed elements:

- The edge e
- The edge E
- The relative distance and direction from v' to f (i.e. the location of both v' and f has not yet been determined, but the location of each one relative to the other is fixed: v' and f are both in T_j, which is perturbed as a rigid object).

For convenience, we transform the coordinate system so that e is in the x axis, and E is in a plane parallel to the xy plane. See Figure B.23.

Let us look at the plane PL_θ that passes through e and has an angle of θ relative to the xy plane. Figure B.24 shows a cross section of the yz plane showing PL and θ.

We start by looking at connectors H which are in the plane PL_θ. v' creates such connectors if it moves only in two dimensions, inside PL_θ.

Let s_θ be the intersection segment created by PL_θ and f, with coordinates relative to v'. The movement of v' moves f as well, and therefore moves s_θ. s_θ has the same length and orientation as long as v' moves in the PL_θ plane.

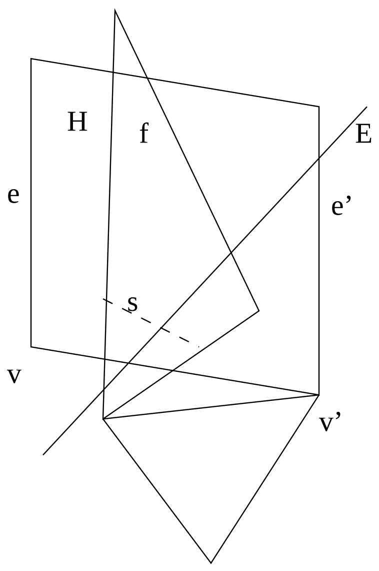


Figure B.34: The location of an edge E in M_{j-1} relative to a segment s created by a facet f in T_j and a facet H in $Con(T_j)$.

Let i_θ be a point on E. i_θ is defined to be the intersection point of E and PL_θ.

We move v' so that s_θ intersects E. The geometric location of v' that induces an intersection of E and s_θ is the segment $\{i_\theta\} \ominus s_\theta$. See Figure B.35. Next we refer to the fact that our interest is not only in E, but in the

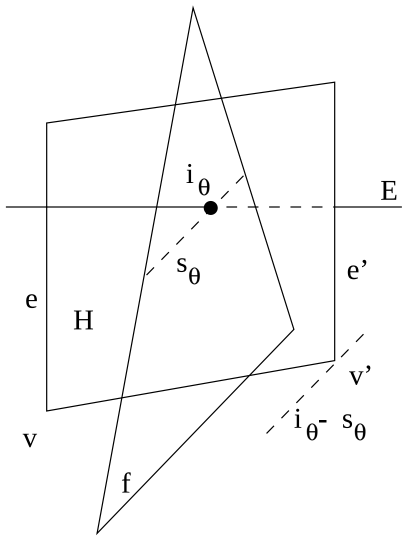


Figure B.35: v' induces an intersection of E and s_θ if it moves along the segment $\{i_\theta\} \ominus s_\theta$.

ε-expanded edge around E. An intersection of an *ε-expanded edge* with a plane is an intersection of a cylinder with a plane, which produces an ellipse.

We change here the notation i_θ: It does not denote a point anymore; it denotes a cylinder cross section. The geometric location of v' that induces an intersection of $Exp(\varepsilon, E)$ and s_θ is therefore the planar Minkowski sum $i_\theta \ominus s_\theta$. See Figure B.36.

Let the possible range for θ be $[\alpha, \beta]$. (α and β are computed later).

For each $\theta \in [\alpha, \beta]$ we get a different planar Minkowski difference $i_\theta \ominus s_\theta$. The range $[\alpha, \beta]$ is continuous, which implies that the areas $i_\theta \ominus s_\theta$ are continuous and their union forms a three-dimensional volume. See Figure B.27.

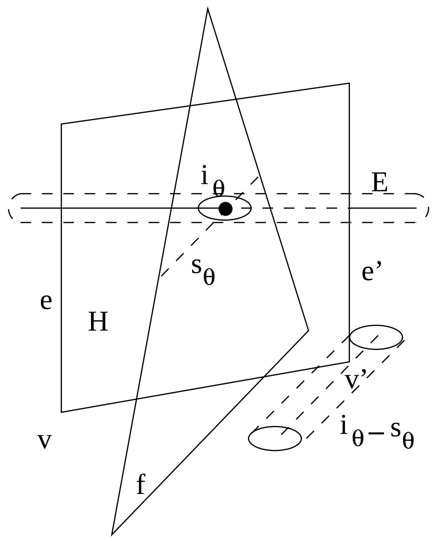


Figure B.36: v' induces an intersection of $Exp(\varepsilon, E)$ and s_θ if it moves along the planar Minkowski difference $s_\theta \ominus i_\theta$.

Let $Loci(E, f, v')$ denote the forbidden loci for v', as analyzed above, with E, f as defined above. Then

$F_{glob\text{-}e\text{-}s\text{-}5} = \{Loci(E, f, v') \mid E \in Set(edge, M_{j-1}),\ f \in Set(facet, T_j),\ v' \in Set(boundary\_vertex, M_{j-1})\}$

Volume Computation

Defining the endpoints of E We use the terms 'leftmost' and 'rightmost' as seen when looking in the negative x direction. Let (x_0, y_0, z_0) be the offset of the leftmost vertex of E from v. Let $(a, b, 0)$ be a unit direction vector of E. (The direction in the z axis is zero since E is in the xy plane). The worst case E is the longest E. Therefore we choose an E of length $\mathcal{L}$. See Figure B.37.

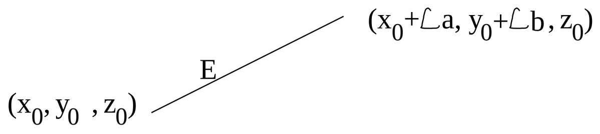


Figure B.37: Endpoints of E (as offset from v)

Computing the Range $[\alpha, \beta]$ Let us find the range $[\alpha, \beta]$ for connectors H. We are interested only in connectors that intersect the edge E.

The angle α is defined when the rightmost end of E touches the connector H (Figure B.38(a)) and the angle β is defined when the leftmost end of E touches the connector H (Figure B.38(b)). It is immediate that

$\tan\alpha = \frac{z_0}{y_0+\mathcal{L}b}$, $\sin\alpha = \frac{z_0}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}}$, $\cos\alpha = \frac{y_0+\mathcal{L}b}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}}$

$\tan\beta = \frac{z_0}{y_0}$, $\sin\beta = \frac{z_0}{\sqrt{y_0^2+z_0^2}}$, $\cos\beta = \frac{y_0}{\sqrt{y_0^2+z_0^2}}$

Considering the Location and Size of f We now know the range of the angle θ, but we do not know where in this range $Exp(\varepsilon, E)$ intersects f. Since we are looking for a worst case, we can always find such a facet f which intersects $Exp(\varepsilon, E)$ for every $\theta \in [\alpha, \beta]$. Therefore we assume that the range for θ is not smaller than $[\alpha, \beta]$.

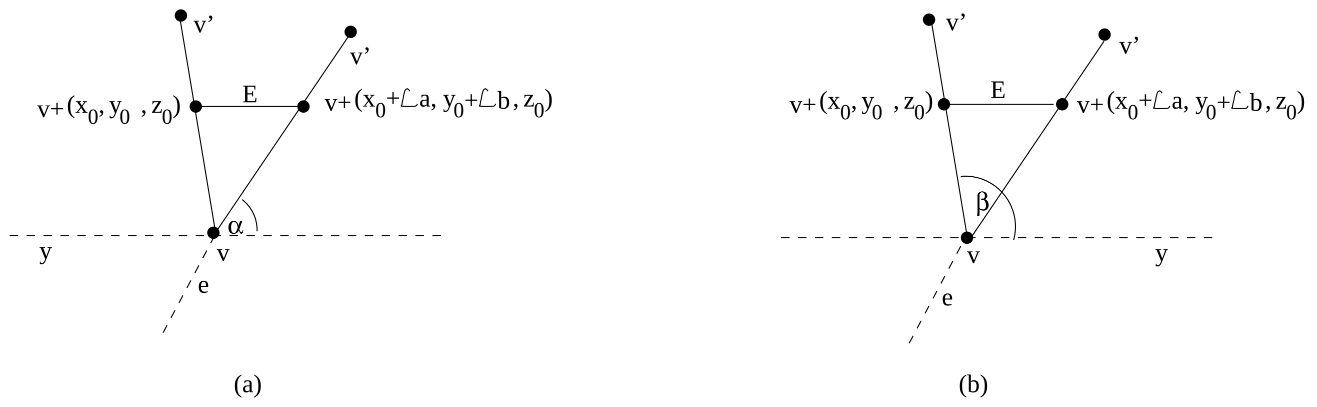


Figure B.38: A projection on the yz plane. (a) Defining the angle α. (b) Defining the angle β.

Computing the Intersection Ellipse of $Exp(\varepsilon, E)$ and f

Claim B.2 *For every angle $\theta \in [\alpha, \beta]$, the length of the major axis of the ellipse i_θ, which is the intersection of $Exp(\varepsilon, E)$ and PL_θ, is* $\frac{2\varepsilon}{\sin\theta \cdot b}$

Proof:
Identical to the one in Claim B.1.

■

Discussing the Area $i_\theta \ominus s_\theta$ The Minkowski difference is illustrated in Figure B.39. For the same reasons as in Section B.3.3.4, we consider a maximum area of $\frac{2\varepsilon}{\sin\theta\cdot b} \cdot (length(s_\theta) + 2\varepsilon)$.

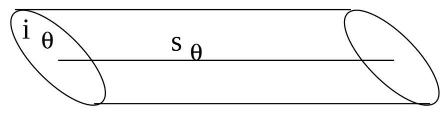


Figure B.39: The Minkowski difference $i_\theta \ominus s_\theta$.

Bounding $length(s_\theta) + 2\varepsilon$ For the same reasons as in Section B.3.3.4, we consider a length of δ_2.

Computing the Volume that Induces Degeneracies For each angle θ, s_θ is 'dragged' along the point i_θ. For the same reasons as in Section B.3.3.4, we use the following integral.

$$\begin{aligned} Volume(\,Loci(E, f, v')\,) &\leq \int_\alpha^\beta \frac{2\varepsilon}{\sin\theta \cdot b} \frac{\delta_2^2}{2} d\theta \\ &= \frac{\varepsilon\delta_2^2}{b} \ln(\csc\theta - \cot\theta) \,|_\alpha^\beta \\ &= \frac{\varepsilon\delta_2^2}{b} \ln \frac{\sqrt{y_0^2 + z_0^2} - y_0}{\sqrt{(y_0 + \mathcal{L}b)^2 + z_0^2} - (y_0 + \mathcal{L}b)} \end{aligned}$$

We next look for an upper bound for the volume.

$Volume(\,Loci(E, f, v')\,) = \frac{\varepsilon\delta_2^2}{b} \ln \frac{\sqrt{y_0^2+z_0^2}-y_0}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)} = \frac{\varepsilon\delta_2^2}{b} \ln(1 + \frac{\sqrt{y_0^2+z_0^2}-\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}+\mathcal{L}b}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)})$

Let us look at the triangle depicted in Figure B.33. According to triangle inequality, we get $\sqrt{y_0^2 + z_0^2} - \sqrt{(y_0 + \mathcal{L}b)^2 + z_0^2} \leq \mathcal{L}b$. Therefore $Volume \leq \frac{\mathcal{L}\varepsilon\delta_2^2}{\mathcal{L}b} \ln(1 + \frac{2}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)}\mathcal{L}b)$.

$\frac{2}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)} > 0$ and according to the inequality $\ln(1+x) \le x$ (for any $x > 0$) we get that $Volume(\,Loci(E,f,v')\,) \le \frac{\varepsilon\delta_2^2}{b} \cdot \frac{2}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)}\mathcal{L}b = 2\mathcal{L}\delta_2^2\frac{\varepsilon}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)}$

$d(v,E) > \rho$ and therefore $\frac{1}{\sqrt{(y_0+\mathcal{L}b)^2+z_0^2}-(y_0+\mathcal{L}b)} < \frac{1}{\rho}$, so we get

$$Volume(\,Loci(E,f,v')\,) \le 2\mathcal{L}\delta_2^2\frac{\varepsilon}{\rho}$$

We are looking for the product of:

- The maximum number of facets in a polyhedral terrain, $2V-2$.
- The maximum number of intr2 that a facet might intersect, $2\binom{K}{2}$.
- $Volume(\,Loci(E,f,v')\,)$

$$Volume(F_{glob\text{-}e\text{-}s\text{-}5}) \le 4VK^2\delta_2^2\mathcal{L}\frac{\varepsilon}{\rho}$$

B.3.3.6 Volumes Induced by a Segment Created by an Intersection of one Facet from M_{j-1} and One Facet from T_j Relative to an Edge in $Con(T_j)$

Geometric Analysis Let S_6 be the set of intersection segments created by a facet in M_{j-1} and a facet in T_j. For every segment s in S_6 and for every edge e'' in $Con(T_j)$, we would like to have $d(s,e'') > \varepsilon$.

Let e, e', v, v', H be as defined in Section B.1.2.1. Let e'' be the edge connecting v and v'. Let f be a facet in M_{j-1} and let F be a facet in T_j. A degeneracy occurs whenever the intersection of f and F intersects the ε-expanded edge around e''. See Figure B.40.

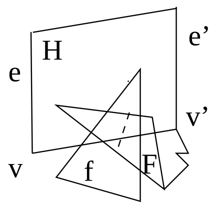


Figure B.40: Intersection of an edge (v,v') in $Con(T_j)$, with a segment created by a facet f in M_{j-1} and a facet F in T_j.

Fixed elements:

- The edge e
- The facet f
- The relative distance and direction from v' to F (i.e. the location of both v' and F has not yet been determined, but the location of each one relative to the other is fixed: v' and F are both in T_j, which is perturbed as a rigid object).

For convenience, we transform the coordinate system so that f is in the xy-plane, and F is parallel to the x-axis (i.e., its normal is parallel to the yz-plane). See Figure B.41.

Whenever v' moves in a fixed height (i.e., it moves in a plane parallel to the xy-plane), the intersection segment of F and f is contained in a fixed segment s of F. Let s be the intersection segment in F for a certain height of v'. For the same reason the intersection of $Exp(\varepsilon, F)$ and f is fixed. See Figure B.42.

In a certain height of v', the collections of forbidden locations are those that induce an intersection of the edge e'' with the intersection of $Exp(\varepsilon, F)$ and f (depicted in Figure B.43). The movements of v' cause movements of F too. The forbidden area for v' for that certain height is similar to the intersection area, only bigger.

The union of the planar forbidden areas for every possible height of v' induces the forbidden volume for v'.

Let $Loci(F,f,x,v')$ denote the forbidden loci for v', with E, f as defined above, where x is the other end of edge e'', i.e., v or u. Then

$$F_{glob\text{-}e\text{-}s\text{-}6} = \{Loci(F,f,v') \,|\, F \in Set(facet, T_j),\ f \in Set(facet, M_{j-1}),\ v' \in Set(boundary\_vertex, M_{j-1})\}$$

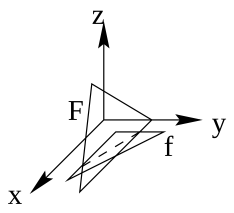


Figure B.41: A transformation of the coordinate system placing f and F in convenient positions.

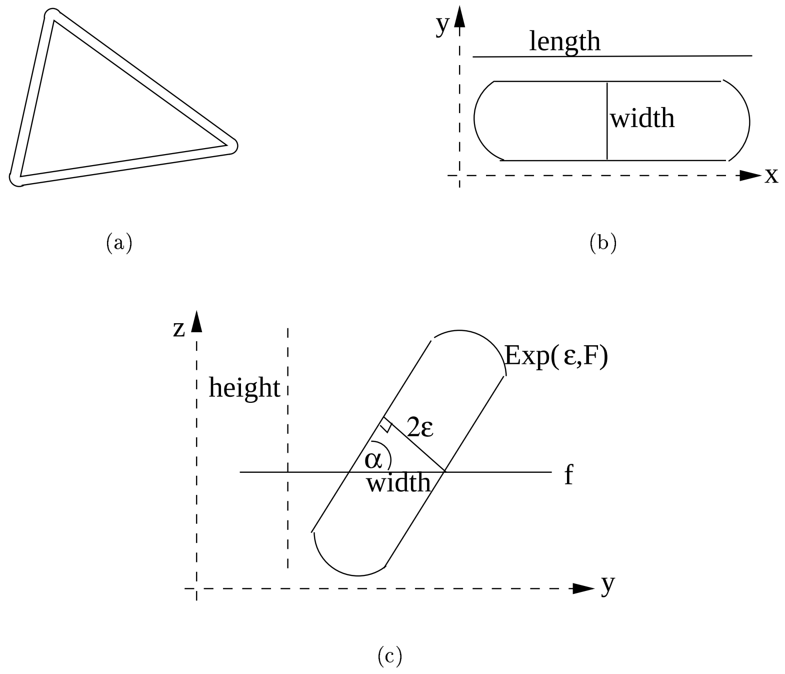


Figure B.42: A two-dimensional illustration of $Exp(\varepsilon, F)$, (b) the intersection of $Exp(\varepsilon, F)$ and f, shown in a cross section of the xy-plane, (c) $Exp(\varepsilon, F)$, shown in a cross section of the yz-plane.

Volume Computation The illustrations in this paragraph are given for $e'' = (v, v')$. The same computations hold for the edge (u, v').

We define *length* and *width* to be the maximum length and width, respectively, of the intersection of $Exp(\varepsilon, F)$ and f. See Figure B.42(b). It is immediate that $length = \mathcal{L}$.

The maximum width of the intersection of $Exp(\varepsilon, F)$ and f (i.e., *width*) is as follows. See also Figure B.42(c). Let $(0, b, c)$ be the normal of F. In a cross section of the yz-plane, let α be the angle between F and f. Since f is in the xy-plane, α is also the angle between F and the xy-plane. According to Figure B.42(c) $\sin\alpha = \frac{2\varepsilon}{width}$, and according to Figure B.44, $\sin\alpha = b$. Therefore $width = \frac{2\varepsilon}{b}$.

We define *height* to be the maximum vertical height of facet F. See also Figure B.42(c).

$\frac{height}{\mathcal{L}} = \sin\alpha \Longrightarrow height = \mathcal{L}b$

Bounding the Movement of v' Suppose v' is moving now horizontally in a certain height. By abuse of notation, let $D(ent1, ent2)$ define the vertical distance between two entities (i.e., the difference between their z coordinates). Let $a = D(v', s) = D(v', f)$ and let $k = D(v, s) = D(v, f)$. Let l be the length of the intersection segment s of F and f. Let r be the intersection point of e'' and s when e'' is vertical. Let l' be the length of the portion of s between its left endpoint (as seen when looking in the positive y direction) and r. See Figure B.45(a). Our goal is to find w, the length of movement of v' so that s is created. Let q be the intersection point of a vertical line going through v and the virtual horizontal line on which v' is moving. Let w_1 be the distance

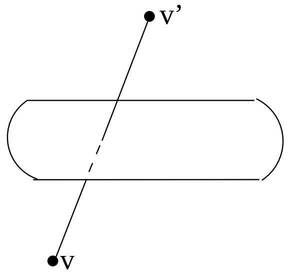


Figure B.43: In a certain height of v', the collections of forbidden locations are those that induce an intersection of the edge $e'' = (v, v')$ with the intersection of $Exp(\varepsilon, F)$ and f .

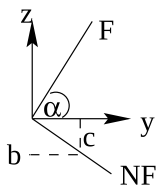


Figure B.44: A cross section of the yz-plane, showing the angle α between F and the xy-plane (which is also the angle between F and f). NF represents the normal of F.

between the rightmost position of v' and q. Let w_2 be the distance between the leftmost position of v' and q. See Figure B.45(b).

Due to triangles similarity, we get:
$\frac{w_1}{a+k} = \frac{l'}{a}$ and $\frac{w_2}{a+k} = \frac{l-l'}{a}$. Therefore $w = w_1 + w_2 = \frac{a+k}{a} \cdot l$. Since s does not keep its maximum length during the movement of F, we should fix the formula into an inequality: $w \leq \frac{a+k}{a} \cdot l$.

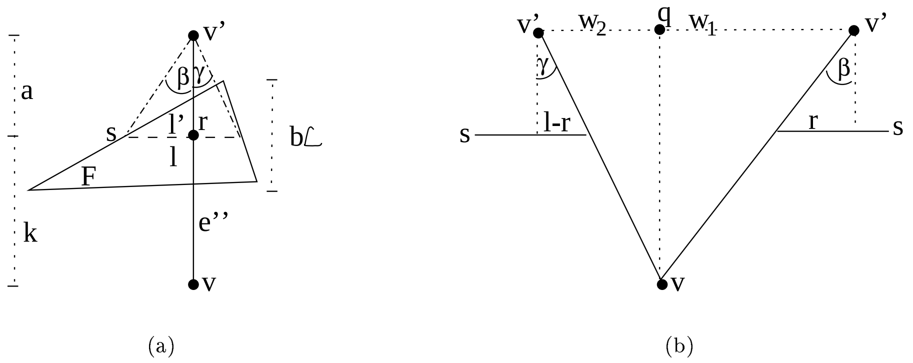


Figure B.45: (a) Defining vertical distances, (b) v' drags F as it moves, and therefore drags s. The drawing shows the domain where $e'' = (v, v')$ intersects s, considering F.

The former computation gave us the terms for e'' to intersect F. Now we compute the terms for e'' to intersect f. The symbols a, k, l, l', w_1, w_2 and w remain as before. The movement of v' does not change the position of f, so the computation is different. See Figure B.46. Due to triangles similarity, we get: $\frac{w_1}{a+k} = \frac{l-l'}{k}$ and $\frac{w_2}{a+k} = \frac{l'}{k}$. Therefore $w = w_1 + w_2 = \frac{a+k}{k} \cdot l$. Since s does not keep its maximum length during the move of v' (which induces a movement of F), we should fix the formula into an inequality: $w \leq \frac{a+k}{k} \cdot l$.

We can now compute a bound for w. Recall that w is the length of movement of v' so that s is created. Notice that $\min(\frac{k}{a}, \frac{a}{k}) \leq 1$.

$length = w \leq \min(\frac{a+k}{a} \cdot l, \frac{a+k}{k} \cdot l) = l \cdot \min(1 + \frac{k}{a}, 1 + \frac{a}{k}) = l \cdot (1 + \min(\frac{k}{a}, \frac{a}{k})) \leq 2l \leq 2\mathcal{L}$

Next we compute a bound for the width of the movement of v'. We use the same inequality that was given

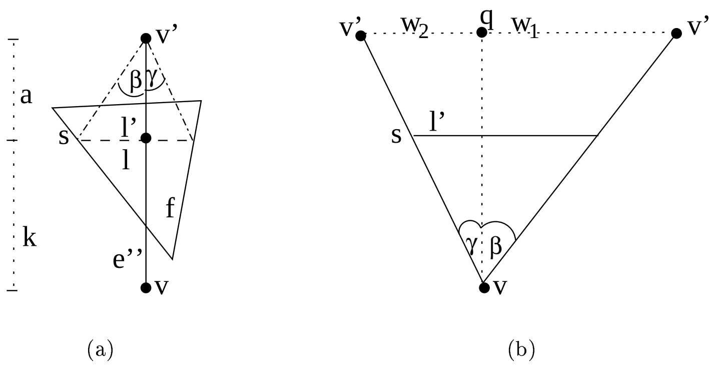


Figure B.46: (a) Same as Figure B.45(a), but this time f is drawn and F is not drawn. (b) The domain where $e'' = (v, v')$ intersects s, considering f.

before, $w \leq \frac{a+k}{a} \cdot l$, and replace l by $\frac{2\varepsilon}{b}$.

$width \leq \frac{a+k}{a} \cdot \frac{2\varepsilon}{b} \leq \frac{2\delta_2\varepsilon}{\rho b}$

Altogether, we have:

$Volume(\,Loci(F, f, v')\,) \leq width \cdot length \cdot height \leq 4\delta_2\mathcal{L}^2\frac{\varepsilon}{\rho}$

We are looking for the product of:

- The maximum number of facets in a polyhedral terrain, $2V - 2$.
- The maximum number of intr2 that a facet might intersect, $2\binom{K}{2}$.
- $Volume(\,Loci(F, f, v')\,)$

$$Volume(F_{glob\text{–}e\text{–}s\text{–}6}) \leq 8VK^2\delta_2\mathcal{L}^2\frac{\varepsilon}{\rho}$$

B.3.4 Removing Degeneracy of Type *facet-intr3*

B.3.4.1 Volumes Induced by the Location of Two Polyhedral Terrains Relative to Each Other

Geometric Analysis For every facet f in T_j, and for every intr3 i in M_{j-1}, we would like to have $d(f, i) > \varepsilon$. Therefore

$$F_{glob\text{–}f\text{–}i\text{–}1} = Set(\,intr3,\, M_{j-1}) \ominus Set(\,facet,\, T_j) \oplus B(0, \varepsilon)$$

Volume Computation We are looking for the product of:

- The maximum number of facets in a polyhedral terrain, $2V - 2$.
- The maximum number of intr3 that a facet might intersect, $\binom{K}{3}$.
- The volume of an ε-*expanded triangle.*

$$\begin{aligned} Volume(F_{glob\text{–}f\text{–}i\text{–}1}) &\leq (2V-2)\binom{K}{3}\varepsilon(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\varepsilon + \frac{10}{3}\pi\varepsilon^2) \\ &\leq \frac{1}{3}VK^3\varepsilon(\mathcal{L}^2 + \frac{3}{2}\mathcal{L}\pi\varepsilon + \frac{10}{3}\pi\varepsilon^2) \end{aligned}$$

B.3.4.2 Volumes Induced by the Location of an Intr3 in M_{j-1} Relative to a Facet in $Con(T_j)$

For every intr3 i in M_{j-1}, and for every connector H created by a perturbation of T_j, we would like to have $d(i,H) > \varepsilon$.

The analysis and computation are exactly as in Section B.3.1.4, when replacing w by i.

Therefore the volume of $F_{glob\text{–}f\text{–}i\text{–}2}$ is the product of:

- The maximum number of vertices on the boundary of a polyhedral terrain, N.
- The maximum number of intr3 that a facet might intersect, $\binom{K}{3}$.
- $Volume(\,Slice(i, v, v', \delta_2)\,)$.

$$Volume(F_{glob\text{–}f\text{–}i\text{–}2}) \leq \frac{2}{9}K^3 N \frac{\varepsilon}{\sqrt{\rho^2 - \varepsilon^2}}\delta_2^3$$

B.3.4.3 Volumes Induced by an Intr3 Created by an Intersection of Two Facets from M_{j-1} and One Facet From T_j Relative to a Facet in $Con(T_j)$

For every intr3 i created by two facets from M_{j-1} and one facet from T_j, and for every connector H created by a perturbation of T_j, we would like to have $d(i,H) > \varepsilon$.

The analysis and computation are exactly as in Section B.3.3.5. The only change is that instead of E in Section B.3.3.5, we have here a segment s created by two facets from M_{j-1}.

According to Section B.3.3.5, $Volume(\,Loci(s, f, v')\,) \leq 2\mathcal{L}\delta_2^2\frac{\varepsilon}{\rho}$.

$$F_{glob\text{–}f\text{–}i\text{–}3} = \{Loci(s, f, v') \mid s \in Set(segment, M_{j-1}),\ f \in Set(facet, T_j)\}$$

We are looking for the product of:

- The maximum number of facets in a polyhedral terrain, $2V - 2$.
- The maximum number of intr3 that a facet might intersect, $\binom{K}{3}$.
- $Volume(\,Loci(s, f, v')\,)$.

$$Volume(F_{glob\text{–}f\text{–}i\text{–}3}) \leq \frac{2}{3}VK^3\delta_2^2\mathcal{L}\frac{\varepsilon}{\rho}$$

B.4 Computing a Bound on δ

B.4.1 General

Outline After computing all forbidden loci, we obtain a list of volumes, where each volume is a function of $\varepsilon, \rho, \lambda, K, L, N, V, \delta_1$ and δ_2. We bound every volume by a term of the form $c_1\pi\varepsilon^{0.5}KL^3V$ for the local step, and $c_2\pi\varepsilon^{0.5}K^3L^3V$ for the global step, where c_1, c_2 are constants. 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 $\delta_2 > (\frac{3}{2}c_2)^{\frac{1}{3}}\varepsilon^{\frac{1}{6}}KLV^{\frac{1}{3}}$.

Bounding Rules We assume $K \geq 10$, $V \geq 10$, $L \geq 10$, $N \leq V$, $\delta_1 \leq L$, $\delta_2 \leq L$ and $\varepsilon \leq 10^{-4}$, and assign $\rho = \varepsilon^{0.5}$ and $\lambda = \rho^{0.5}$.

Then we bound every item in a computed volume according to the rules in Table B.3.

Origin	Bound by	Notes/Justification
1	$\frac{1}{3}\pi$	Only if π does not appear in the formula ($1 < \frac{1}{3}\pi$)
$\varepsilon, \varepsilon^2, \varepsilon^3$	$10^{-2}\varepsilon^{0.5}, 10^{-6}\varepsilon^{0.5}, 10^{-10}\varepsilon^{0.5}$	$\varepsilon \leq 10^{-4}$
ρ, ρ^2, ρ^3	$\varepsilon^{0.5}, 10^{-2}\varepsilon^{0.5}, 10^{-4}\varepsilon^{0.5}$	$\rho = \varepsilon^{0.5}$
λ^3	$10^{-1}\varepsilon^{0.5}$	$\lambda = \rho^{0.5}$
$\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}$	$\sqrt{1.0001}\varepsilon^{0.5}$	$\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}} = \sqrt{\frac{\varepsilon}{\varepsilon-\varepsilon^2}}\varepsilon^{0.5} = \sqrt{\frac{1}{1-\varepsilon}}\varepsilon^{0.5} \leq \sqrt{1.0001}\varepsilon^{0.5}$
$\frac{\rho^2}{\lambda^2-\rho^2}$	$1.011\varepsilon^{0.5}$	$\frac{\rho^2}{\lambda^2-\rho^2} = \frac{\varepsilon^{0.5}}{\varepsilon^{0.5}-\varepsilon}\varepsilon^{0.5} = \frac{1}{1-\varepsilon^{0.5}}\varepsilon^{0.5} \leq 1.011\varepsilon^{0.5}$
K, K^2	$10^{-2}K^3$, $10^{-1}K^3$	Only for the computation of δ_2 ($K \geq 10$)
N	V	$N \leq V$
1	$10^{-1}V$	Only if V does not appear in the formula ($V \geq 10$)
L, L^2	$10^{-2}L^3$, $10^{-1}L^3$	$L \leq 10$
1	$10^{-3}L^3$	Only if L does not appear in the formula ($L \leq 10$)
δ_1, δ_2	L	$\delta_1 \leq L$, $\delta_2 \leq L$
$\mathcal{L}$	$3L$	$\mathcal{L} := L + 2\delta_1$

Table B.3: Bounding rules

B.4.2 Computing a Bound on δ_1

In Table B.4 we compute the bound for every volume computed for the local step. The total sum is bounded by $167.22\pi\varepsilon^{0.5}KL^3V$ and therefore we conclude that

$$\delta_1 > 6.31\,\varepsilon^{1/6}K^{1/3}LV^{1/3}$$

Volume	Bounded by	
$\frac{4}{3}\pi K\lambda^3$	$\frac{4}{3}\pi(0.1\varepsilon^{0.5})K(10^{-3}L^3)(10^{-1}V)$	$\leq 2\cdot 10^{-5}\cdot\pi\varepsilon^{0.5}KL^3V$
$2\pi KV\frac{\rho^2}{\lambda^2-\rho^2}\mathcal{L}^3$	$2\pi(1.011\varepsilon^{0.5})K(27L^3)V$	$\leq 54.6\cdot\pi\varepsilon^{0.5}KL^3V$
$2\pi K\rho^2(\mathcal{L}+\frac{4}{3}\rho)$	$2\pi(10^{-2}\varepsilon^{0.5})K(3\cdot 10^{-2}L^3)(10^{-1}V)+$ $+2\pi(\frac{4}{3}10^{-4}\varepsilon^{0.5})K(10^{-3}L^3)(10^{-1}V)$	$\leq (6\cdot 10^{-5}+3\cdot 10^{-8})\cdot\pi\varepsilon^{0.5}KL^3V$
$4.5\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}KV\mathcal{L}^3$	$4.5(\frac{1}{3}\pi)\sqrt{1.0001}\varepsilon^{0.5}K(27L^3)V$	$\leq 40.51\cdot\pi\varepsilon^{0.5}KL^3V$
$\varepsilon K(\mathcal{L}^2+\frac{3}{2}\pi\mathcal{L}\varepsilon+\frac{10}{3}\pi\varepsilon^2)$	$(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})K(9\cdot 10^{-1}L^3)(10^{-1}V)+$ $+\frac{3}{2}\pi(10^{-6}\varepsilon^{0.5})K(3\cdot 10^{-2}L^3)(10^{-1}V)+$ $+\frac{10}{3}\pi(10^{-10}\varepsilon^{0.5})K(10^{-3}L^3)(10^{-1}V)$	$\leq (3\cdot 10^{-4}+5\cdot 10^{-9}+4\cdot 10^{-14})\cdot\pi\varepsilon^{0.5}KL^3V$
$8\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}KV\mathcal{L}^3$	$8(\frac{1}{3}\pi)\sqrt{1.0001}\varepsilon^{0.5}K(27L^3)V$	$\leq 72.1\cdot\pi\varepsilon^{0.5}KL^3V$

Table B.4: Bounding the volumes induced by the local step, in order to reach a uniform formula.

B.4.3 Computing a Bound on δ_2

In Table B.5 we compute the bound for every volume computed for the global step. The total sum is bounded by $6.54\pi\varepsilon^{0.5}K^3L^3V$ and therefore we conclude that

$$\delta_2 > 2.15\,\varepsilon^{1/6}KLV^{1/3}$$

Volume	Bounded by	
Related to *vertex-facet* degeneracy		
$6VK\varepsilon(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\varepsilon+\frac{10}{3}\pi\varepsilon^2)$	$6(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(9\cdot 10^{-1}L^3)V+$ $+6\cdot\frac{3}{2}\pi(10^{-6}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+6\cdot\frac{10}{3}\pi(10^{-10}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (2\cdot 10^{-4}+$ $+3\cdot 10^{-9}+$ $+2\cdot 10^{-14})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$KV\varepsilon(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\varepsilon+\frac{10}{3}\pi\varepsilon^2)$	$(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(9\cdot 10^{-1}L^3)V+$ $+\frac{3}{2}\pi(10^{-6}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+\frac{10}{3}\pi(10^{-10}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (3\cdot 10^{-5}+$ $+5\cdot 10^{-10}+$ $+4\cdot 10^{-15})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$2\pi NK\rho^2(\mathcal{L}+\frac{4}{3}\rho)$	$2\pi(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+2\cdot\frac{4}{3}\pi(10^{-4}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (6\cdot 10^{-6}+$ $+3\cdot 10^{-9})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$2\pi VK\rho^2(\mathcal{L}+\frac{4}{3}\rho)$	$2\pi(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+2\cdot\frac{4}{3}\pi(10^{-4}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (6\cdot 10^{-6}+$ $+3\cdot 10^{-9})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$6KV\rho(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\rho+\frac{10}{3}\pi\rho^2)$	$6(\frac{1}{3}\pi)(\varepsilon^{0.5})(10^{-2}K^3)(9\cdot 10^{-1}L^3)V+$ $+6\cdot\frac{3}{2}\pi(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+6\cdot\frac{10}{3}\pi(10^{-4}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (1.8\cdot 10^{-2}+$ $+3\cdot 10^{-5}+$ $+2\cdot 10^{-8})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$KN\rho(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\rho+\frac{10}{3}\pi\rho^2)$	$(\frac{1}{3}\pi)(\varepsilon^{0.5})(10^{-2}K^3)(9\cdot 10^{-1}L^3)V+$ $+\frac{3}{2}\pi(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+\frac{10}{3}\pi(10^{-4}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (3\cdot 10^{-3}+$ $+5\cdot 10^{-6}+$ $+4\cdot 10^{-9})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$4KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$	$4(\frac{1}{3}\pi)\sqrt{1.0001}\varepsilon^{0.5}(10^{-2}K^3)L^3V$	$\leq 1.4\cdot 10^{-2}\cdot\pi\varepsilon^{0.5}K^3L^3V$
$4KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$	$4(\frac{1}{3}\pi)\sqrt{1.0001}\varepsilon^{0.5}(10^{-2}K^3)L^3V$	$\leq 1.4\cdot 10^{-2}\cdot\pi\varepsilon^{0.5}K^3L^3V$
Related to *edge-edge* degeneracy		
$12KV\varepsilon(\mathcal{L}^2+\pi\mathcal{L}\varepsilon+2\pi\varepsilon^2)$	$12(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})(10^{-2}K^3)(9\cdot 10^{-1}L^3)V+$ $+12\pi(10^{-6}\varepsilon^{0.5})(10^{-2}K^3)(3\cdot 10^{-2}L^3)V+$ $+12\cdot 2\pi(10^{-10}\varepsilon^{0.5})(10^{-2}K^3)(10^{-3}L^3)V$	$\leq (4\cdot 10^{-4}+$ $+4\cdot 10^{-9}+$ $+3\cdot 10^{-14})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$\frac{8}{3}KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\delta_2^3+(\mathcal{L}+\delta_2)^3)$	$\frac{8}{3}(\frac{1}{3}\pi)(\sqrt{1.0001}\varepsilon^{0.5})(10^{-2}K^3)(65L^3)V$	$\leq 5.78\cdot 10^{-1}\cdot\pi\varepsilon^{0.5}K^3L^3V$
$\frac{8}{3}KN\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\delta_2^3+(\mathcal{L}+\delta_2)^3)$	$\frac{8}{3}(\frac{1}{3}\pi)(\sqrt{1.0001}\varepsilon^{0.5})(10^{-2}K^3)(65L^3)V$	$\leq 5.78\cdot 10^{-1}\cdot\pi\varepsilon^{0.5}K^3L^3V$
Related to *edge-segment* degeneracy		
$3VK^2\varepsilon(\mathcal{L}^2+\pi\mathcal{L}\varepsilon+2\pi\varepsilon^2)$	$3(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})(10^{-1}K^3)(9\cdot 10^{-1}L^3)V+$ $+3\pi(10^{-6}\varepsilon^{0.5})(10^{-1}K^3)(3\cdot 10^{-2}L^3)V+$ $+3\cdot 2\pi(10^{-10}\varepsilon^{0.5})(10^{-1}K^3)(10^{-3}L^3)V$	$\leq (9\cdot 10^{-4}+$ $+9\cdot 10^{-9}+$ $+6\cdot 10^{-14})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$2VK^2\varepsilon(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\varepsilon+\frac{10}{3}\pi\varepsilon^2)$	$2(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})(10^{-1}K^3)(9\cdot 10^{-1}L^3)V+$ $+2\cdot\frac{3}{2}\pi(10^{-6}\varepsilon^{0.5})(10^{-1}K^3)(3\cdot 10^{-2}L^3)V+$ $+2\cdot\frac{10}{3}\pi(10^{-10}\varepsilon^{0.5})(10^{-1}K^3)(10^{-3}L^3)V$	$\leq (6\cdot 10^{-4}+$ $+9\cdot 10^{-9}+$ $+7\cdot 10^{-14})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$\frac{2}{3}K^2N\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}(\delta_2^3+(\mathcal{L}+\delta_2)^3)$	$\frac{2}{3}(\frac{1}{3}\pi)(\sqrt{1.0001}\varepsilon^{0.5})(10^{-1}K^3)(65L^3)V+$	$\leq 1.446\cdot\pi\varepsilon^{0.5}K^3L^3V$
$\frac{4}{3}K^2N\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$	$\frac{4}{3}(\frac{1}{3}\pi)(\sqrt{1.0001}\varepsilon^{0.5})(10^{-1}K^3)L^3V$	$\leq 4.45\cdot 10^{-2}\cdot\pi\varepsilon^{0.5}K^3L^3V$
$3VK^2\delta_2^2\mathcal{L}\frac{\varepsilon}{\rho}$	$3(\frac{1}{3}\pi)\varepsilon^{0.5}(10^{-1}K^3)(3L^3)V$	$\leq 3\cdot 10^{-1}\cdot\pi\varepsilon^{0.5}K^3L^3V$
$8VK^2\delta_2\mathcal{L}^2\frac{\varepsilon}{\rho}$	$8(\frac{1}{3}\pi)\varepsilon^{0.5}(10^{-1}K^3)(9L^3)V$	$\leq 2.4\cdot\pi\varepsilon^{0.5}K^3L^3V$
$4VK^2\delta_2^2\mathcal{L}\frac{\varepsilon}{\rho}$	$4(\frac{1}{3}\pi)\varepsilon^{0.5}(10^{-1}K^3)(3L^3)V$	$\leq 4\cdot 10^{-1}\cdot\pi\varepsilon^{0.5}K^3L^3V$
Related to *facet-intr3* degeneracy		
$\frac{1}{3}VK^3\varepsilon(\mathcal{L}^2+\frac{3}{2}\mathcal{L}\pi\varepsilon+\frac{10}{3}\pi\varepsilon^2)$	$\frac{1}{3}(\frac{1}{3}\pi)(10^{-2}\varepsilon^{0.5})K^3(9\cdot 10^{-1}L^3)V+$ $+\frac{1}{3}\cdot\frac{3}{2}\pi(10^{-6}\varepsilon^{0.5})K^3(3\cdot 10^{-2}L^3)V+$ $+\frac{1}{3}\cdot\frac{10}{3}\pi(10^{-10}\varepsilon^{0.5})K^3(10^{-3}L^3)V$	$\leq (10^{-3}+$ $+1.5\cdot 10^{-8}+$ $+1.2\cdot 10^{-13})\cdot\pi\varepsilon^{0.5}K^3L^3V$
$\frac{2}{9}K^3N\frac{\varepsilon}{\sqrt{\rho^2-\varepsilon^2}}\delta_2^3$	$\frac{2}{9}(\frac{1}{3}\pi)(\sqrt{1.0001}\varepsilon^{0.5})K^3L^3V$	$\leq 7.41\cdot 10^{-2}\cdot\pi\varepsilon^{0.5}K^3L^3V$
$\frac{2}{3}VK^3\delta_2^2\mathcal{L}\frac{\varepsilon}{\rho}$	$\frac{2}{3}(\frac{1}{3}\pi)\varepsilon^{0.5}K^3(3L^3)V$	$\leq 6.67\cdot 10^{-1}\cdot\pi\varepsilon^{0.5}K^3L^3V$

Table B.5: Bounding the volumes induced by the global step, in order to reach a uniform formula.

Appendix C

Forbidden Loci for Degeneracy of Type *concurrent-three-edges* (Coordinate System Step)

In this appendix we describe and bound the forbidden loci for degeneracy of type *concurrent-three-edges*, as defined in Section 3.7.

Recall that degeneracy of type *concurrent-three-edges* is defined as follows: Let e_1, e_2, e_3 be non-incident distinct edges. Let l be a line intersecting all three edges. The angle between l and the z-direction is less than ω.

Overview of Computation The direction of any line intersecting all three edges e_1, e_2, e_3 should be ω-separated from the z-direction. In this Appendix we show that the endpoints of the direction vectors of all lines intersecting e_1, e_2, e_3 determine a 3D curve. We then normalize that 3D curve into a parametric curve on the unit sphere, denoted by $f(t)$, $t \in [0, 1]$.

Let S be the unit sphere. Let o be the center of S and let $p = f(t_0)$ be a point on the unit sphere for an arbitrary value of $t_0 \in [0, 1]$. Let C be a circle on S such that every line through o and C makes an angle ω with the line through o and p. See Figure C.1(a). We call such a circle the ω-circle of p. The interior of C is the loci of forbidden z-directions with respect to p. In the same way, the Minkowski sum of $f(t)$ and C is the collection of forbidden z-directions with respect to $f(t)$. See Figure C.1(b). We call this Minkowski sum the ω-strip of $f(t)$.

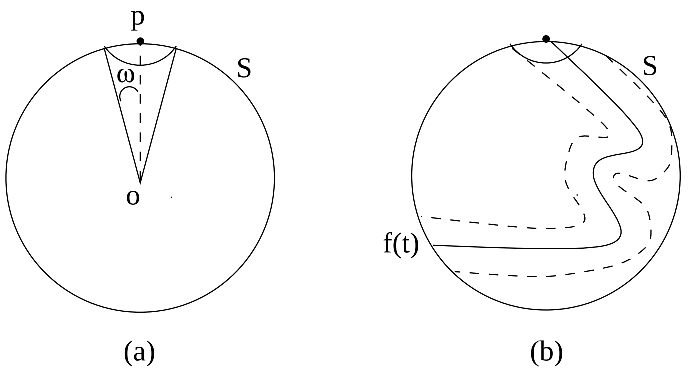


Figure C.1: (a) The ω-circle of p. (b) The ω-strip of $f(t)$.

Computing $f(t)$ Let L_1, L_2, L_3 be the underlying lines of e_1, e_2, e_3 respectively. Let us create a parameterization of those lines as following:

$L_1(t) = a_1 + t \cdot b_1$

$L_2(t) = a_2 + t \cdot b_2$

$L_3(t) = a_3 + t \cdot b_3$

Where a_i, b_i are three-dimensional points, and therefore $a_i = (a_{ix}, a_{iy}, a_{iz})$ and $b_i = (b_{ix}, b_{iy}, b_{iz})$ for $i = 1, 2, 3$. Using a translation and rotation we can obtain $a_1 = (0,0,0)$ and $b_1 = (1,0,0)$, thus $L_1(t) = (t,0,0)$. Using rotation around the x axis we can make sure that L_2 is parallel to the xy-plane and obtain $b_2 = (b_{2x}, b_{2y}, 0)$ thus $L_2(t) = (a_{2x} + t \cdot b_{2x},\ a_{2y} + t \cdot b_{2y},\ a_{2z})$. For every edge the parameterization can be normalized such that $L_i(0)$ is the beginning point of edge e_i and $L_i(1)$ is the ending point of edge e_i ($i = 1, 2, 3$). Therefore $L_i(t)$ for $t \in [0,1]$ is a point in edge e_i.

We next show that for every value $t \in [0,1]$, there exists a line L_4 and values t_2, t_3 such that L_4 intersects $L_1(t)$, and $L_2(t_2)$, $L_3(t_3)$ or their extensions. (t_2, t_3 are not necessarily in $[0,1]$). L_4 passes through the points $L_1(t)$ and $L_3(t_3)$ and therefore can be defined by those two points: $L_4(u) = L_1(t) + u \cdot (L_3(t_3) - L_1(t))$. On the other hand, there exists a value t_4 such that $L_4(t_4)$ is equal to $L_2(t_2)$, and the following equation is obtained:

$$L_1(t) + t_4 \cdot (L_3(t_3) - L_1(t)) = L_2(t_2)$$

For every $t \in [0,1]$, the above is a set of three equations (for x, y, z) with three variables (t_2, t_3, t_4), which is always solvable.

Let P_k denote a polynom of order k in t. It is immediate that $\frac{P_a}{P_b} + \frac{P_c}{P_d} = \frac{P_{\max(a+d,b+c)}}{P_{b+d}}$ and $\frac{P_a}{P_b} \cdot \frac{P_c}{P_d} = \frac{P_{a+c}}{P_{b+d}}$. (We assume the worst-case, where there is no common divisor for the numerator and the denominator).

After solving the three equations[1], we obtain that t_3 is of the form $\frac{P_1}{P_1}$. For every $t \in [0,1]$, the direction of L_4 is $L_3(t_3) - L_1(t) = (a_{3x} + t_3 \cdot b_{3x} - t,\ a_{3y} + t_3 \cdot b_{3y},\ a_{3z} + t_3 \cdot b_{3z})$. Such a vector is of the degrees $(\frac{P_2}{P_1}, \frac{P_1}{P_1}, \frac{P_1}{P_1})$.

We next normalize the direction vector, in order to find its intersection with the unit sphere. In order to normalize the direction vector, we need to divide it by its norm:

$\sqrt{\left(\frac{P_2}{P_1}\right)^2 + \left(\frac{P_1}{P_1}\right)^2 + \left(\frac{P_1}{P_1}\right)^2} = \sqrt{\frac{P_4}{P_2} + \frac{P_2}{P_2} + \frac{P_2}{P_2}} = \sqrt{\frac{P_8}{P_6}}$.

Then the division is:

$\left(\frac{\frac{P_2}{P_1}}{\sqrt{\frac{P_8}{P_6}}}, \frac{\frac{P_1}{P_1}}{\sqrt{\frac{P_8}{P_6}}}, \frac{\frac{P_1}{P_1}}{\sqrt{\frac{P_8}{P_6}}}\right) = \left(\sqrt{\frac{\frac{P_4}{P_2}}{\frac{P_8}{P_6}}}, \sqrt{\frac{\frac{P_2}{P_2}}{\frac{P_8}{P_6}}}, \sqrt{\frac{\frac{P_2}{P_2}}{\frac{P_8}{P_6}}}\right) = \left(\sqrt{\frac{P_{10}}{P_{10}}}, \sqrt{\frac{P_8}{P_{10}}}, \sqrt{\frac{P_8}{P_{10}}}\right).$

By a parameterization of t we obtain a curve on the unit sphere. Let this curve be denoted by $f(t) = (x(t), y(t), z(t))$. The coordinates of $f(t)$ are of the form $\left(\sqrt{\frac{P_{10}}{P_{10}}}, \sqrt{\frac{P_8}{P_{10}}}, \sqrt{\frac{P_8}{P_{10}}}\right)$.

Let a *monotone portion* be a portion of a curve which is monotone both in the x, y, and z axes. Let a *monotone break point* be the point where two consecutive monotone portions meet.

Let us find a bound for the number of monotone portions in $f(t)$. A monotone break point must have at least one of the following characteristics:

- The curve derivative is zero (in at least one axis)
- The curve derivative is undefined (in at least one axis)
- The curve is undefined (in at least one axis)

Derivative of the x coordinate:

$$\left(\sqrt{\frac{P_{10_{top}}}{P_{10_{bottom}}}}\right)' = \frac{1}{2 \cdot \sqrt{\frac{P_{10_{top}}}{P_{10_{bottom}}}}} \cdot \frac{(P_{10_{top}})' \cdot P_{10_{bottom}} - (P_{10_{bottom}})' \cdot P_{10_{top}}}{(P_{10_{bottom}})^2} = \frac{\sqrt{P_{10_{bottom}}}}{2 \cdot \sqrt{P_{10_{top}}}} \cdot \frac{P_{19}}{(P_{10_{bottom}})^2}$$

- The x derivative is zero in the roots of $P_{10_{bottom}}$ and P_{19} .
- The x derivative is undefined in the roots of $P_{10_{bottom}}$ and $P_{10_{top}}$.
- The x coordinate is undefined in the roots of $P_{10_{bottom}}$.

[1] We solved the equations and found the degrees of t_3 using the software *Mathematica*. The *Mathematica* notebook file can be found in http://www.math.tau.ac.il/˜raab/concurrent-three-edges.nb

All together, the maximum number of monotone break points which are derived out of the x coordinate is 39.

Derivative of the y coordinate:

$$\left(\sqrt{\frac{P_8}{P_{10}}}\right)' = \frac{1}{2\cdot\sqrt{\frac{P_8}{P_{10}}}} \cdot \frac{(P_8)'\cdot P_{10} - (P_{10})'\cdot P_8}{(P_{10})^2} = \frac{\sqrt{P_{10}}}{2\cdot\sqrt{P_8}} \cdot \frac{P_{17}}{(P_{10})^2}$$

- The y derivative is zero in the roots of P_{10} and P_{17} .
- The y derivative is undefined in the roots of P_{10} and P_8.
- The y coordinate is undefined in the roots of P_{10}.

All together, the maximum number of monotone break points which are derived out of the y coordinate is 35.

Since the z coordinate has the same degrees as the y coordinate, we obtain that the maximum number of monotone break points which are derived out of the z coordinates is also 35.

Bounding the Length of $f(t)$ Let $len(curve)$ denote the length of a curve. Let $curve_{xy}, curve_{xz}, curve_{yz}$ denote the projections of the curve on the xy, xz, yz planes respectively.

$$\begin{aligned}
len(f(t)) &= \int_0^1 \sqrt{(x'(t))^2 + (y'(t))^2 + (z'(t))^2}dt \\
&= \frac{1}{\sqrt{2}}\int_0^1 \sqrt{(x'(t))^2 + (y'(t))^2 + (x'(t))^2 + (z'(t))^2 + (y'(t))^2 + (z'(t))^2}dt \\
&\leq \frac{1}{\sqrt{2}}\left(\int_0^1 \sqrt{(x'(t))^2 + (y'(t))^2}dt + \int_0^1 \sqrt{(x'(t))^2 + (z'(t))^2}dt + \int_0^1 \sqrt{(y'(t))^2 + (z'(t))^2}dt\right) \\
&= \frac{1}{\sqrt{2}}(len(f(t)_{xy}) + len(f(t)_{xz}) + len(f(t)_{yz}))
\end{aligned}$$

It is obvious that the number of monotone portions of a curve is the number of monotone break points plus one. The number of monotone break points of $f(t)_{xy}$ is bounded by the sum of monotone break points derived out of the x and y directions, which is 39+35=74. The number of monotone portions in $f(t)_{xy}$ is therefore bounded by 75. In the same way, The number of monotone portions in $f(t)_{xz}$ is bounded by 75 and the number of monotone portions in $f(t)_{yz}$ is bounded by 71.

Every projection on one of the trivial planes is bounded in the square $[-1,1]\times[-1,1]$ and therefore the length of a projected monotone portion is bounded by 2+2=4. After multiplying by the number of monotone portions, we conclude that

$$len(f(t)) \leq \frac{1}{\sqrt{2}}\cdot(75+75+71)\cdot 4 < 625.1$$

Bounding the area of the ω-strip around $f(t)$ Let $p_0 = f(t_0)$ be a point on the unit sphere where $t_0 = 0$. We build a collection of circles on the unit sphere by construction: Let C_i be the ω-circle of p_i and let C'_i be the 2ω-circle of p_i. Let p_{i+1} be the point $f(t_{i+1}) = f(t)\bigcap C_i$ for $t_{i+1} > t_i$. For every j such that $i < j < i+1$, the ω-circle of $f(t_j)$ is inside C'_i. The procedure goes on until $f(t)$ is completely covered by ω-circles, i.e., there is no $t_{i+1} > t_i$ such that $f(t_{i+1}) = f(t)\bigcap C_i$. See Figure C.2.

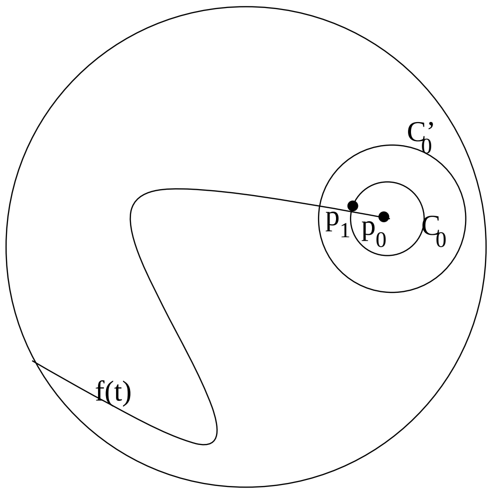
The area $\bigcup C'_i$ bounds the ω-strip of $f(t)$.

The area of one 2ω-circle is (see Figure C.3):

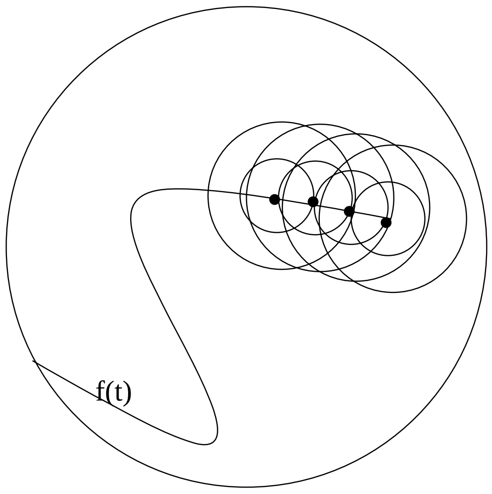
$1 - h = \cos(2\omega)$

$Area(2\omega - circle) = 2\pi h = 2\pi(1-\cos(2\omega)) = 4\pi\sin^2\omega$ The minimum length of a curve starting from the center of an ω-circle and ending at its perimeter is $\frac{\omega}{2\pi}\cdot 2\pi = \omega$. Therefore the number of points p_i is bounded by $\frac{len(f(t))}{\omega} \leq \frac{len(f(t))}{\sin\omega}$.

The area of the ω-strip around $f(t)$ is therefore bounded by $\frac{len(f(t))}{\sin\omega}\cdot 4\pi\sin^2\omega \leq 2501\pi\sin\omega$.



(a)



(b)

Figure C.2: (a) The first steps of the constructive build, (b) the ω-strip of $f(t)$.

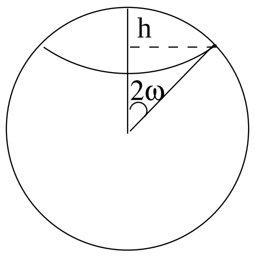


Figure C.3: Computing the area of a 2ω-circle

Bibliography

[1] K. Abdel-Malek and H.J. Yeh. Geometric representation of the swept volume using Jacobian rank-deficiency conditions. *Computer Aided Design*, 29(9):457–468, 1997.

[2] S.A. Abrams and P.K Allen. Computing swept volumes for sensor planning tasks. In *Proc. Image Understanding Workshop*, volume 2, pages 1159–1165, 1994.

[3] 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.

[4] 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.

[5] 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.

[6] D. Blackmore and M.C. Leu. Representation of sweeps and swept volumes via differential equations. In *Proc. of the 1992 NSF Design and Manufacturing Systems Conference*, pages 731–735, 1992.

[7] D. Blackmore, M.C. Leu, and L.P. Wang. The sweep-envelope differential equation algorithm and its application to NC machining verification. *Computer Aided Design*, 29(9):629–637, 1997.

[8] S. Boussac and A. Crosnier. Swept volumes generated from deformable objects: Application to NC verification. In *Proc. of the 1996 IEE Intl. Conference on Robotics and Automation*, pages 1813–1818, 1996.

[9] 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.

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

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

[12] 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.

[13] B.K. Choi. *Surface Modeling for CAD/CAM.* Elsevier Science Publishers Company Inc., New York, NY, 1991.

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

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

[16] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, 1997.

[17] 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.

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

[19] S. Fortune. Robustness issues in geometric algorithms. In M.C. Lin and D. Manocha, editors, *Applied Computational Geometry (Proc. WACG '96)*, volume 1148 of *Lecture Notes Comput. Sci.*, pages 9–14. Springer, 1996.

[20] S. Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 116–125, 1998.

[21] 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.

[22] K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson. *Algorithmic Foundations of Robotics.* A.K. Peters, Wellesley, Massachusetts, 1995.

[23] 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.

[24] D. Greene and F. Yao. Finite resolution computational geometry. In *Proc. 27th IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[25] L.J. Guibas. Implementing geometric algorithms robustly. In M.C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 15–22. Springer, 1996.

[26] 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.

[27] D. Halperin, L. Kavraki, and J.C. Latombe. Robotics. In J.E. Goodman and J. O'Rourke, editors, *CRC Handbook of Discrete and Computational Geometry*, chapter 41, pages 755–778. CRC Press, Inc., Boca Raton, FL, 1997.

[28] D. Halperin and M. Sharir. Arrangements and their applications in robotics: Recent developments. In K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson, editors, *The Algorithmic Foundations of Robotics*, pages 495–511. A.K. Peters, Boston, MA, 1995.

[29] 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.

[30] J. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.* To appear.

[31] C. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, California, 1989.

[32] Z.J. Hu and Z.K. Ling. Swept volumes generated by natural quadric surfaces. *Computers & Graphics*, 20(2):263–274, 1996.

[33] L. Kettner. Designing a data structure for polyheral surfaces. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 146–154, 1998.

[34] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, Masschusetts, 1991.

[35] M.C. Leu, D. Blackmore, L.P. Wang, and K.G. Pak. Implementaition of SDE method to represent cutter swept volumes in 5-axis NC milling. In *Proc. of the SPIE - The Intl. Society for Optical Engineering*, volume 2620, pages 354–364, 1995.

[36] M.C. Leu, H. Park, and K.K. Wang. Geometric representation of translational swept volumes and its applications. *ASME J. Engin. industry*, 108:113–119, 1986.

[37] R.R. Martin and P.C Stephenson. Sweeping of three dimentional objects. *Computer Aided Design*, 22(4):223–234, 1990.

[38] V.J. Milenkovic. Verifiable implementation of geometric algorithms using finite precision arighmetic. *Artificial Intelligence*, 37:377–401, 1988.

[39] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, NY, 1994.

[40] F.P. Preparata. Robustness in geometric algorithms. In M.C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 23–24. Springer, 1996.

[41] S. Raab. Excerpts from M.Sc thesis, in http://www.math.tau.ac.il/~raab/thesis_cites.html. 1998.

[42] 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.

[43] K. Sambandan, K. Kedem, and K.K. Wang. Generalized planar sweeping of polygons. *J. of Manufacturing Systems*, 11(4):246–257, 1992.

[44] 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.

[45] W.J Schroeder, W.E. Lorensen, and S. Linthicum. Implicit modeling of swept surfaces and volumes. In *Proc. Visualization '94*, pages 40–45, 1994.

[46] M. Sharir and P.K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.

[47] H. Shaul and D. Halperin. Partial vs. full vertical decomposition of three-dimensional arrangements. *in preparation.*

[48] J.R. Shewchuk. Adaptive robust floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18:305–363, 1997.

[49] K. Sugihara. On finite-precision representations of geometric objects. *J. Comput. Syst. Sci.*, 39:236–247, 1989.

[50] K. Sugihara and M. Iri. Two design principles of geometric algorithms in finite-precision arithmetic. *Appl. Math. Lett.*, 2(2):203–206, 1989.

[51] W.P. Wang and K.K. Wang. Geometric modelling for swept volume for moving solids. *IEEE Computer Graphics & Applications*, 6(12):18–17, 1986.

[52] 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.

[53] P.G. Xavier. Fast swept-volume distance for robust collision detection. In *Intl. Conf. Robotics and Auto.*, pages 1162–1169, 1997.

[54] C.K. Yap. Symbolic treatment of geometric degeneracies. *J. Symbolic Computation*, 10:349–370, 1990.

[55] C.K. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7:3–23, 1997.