Robust Solid Modeling by Avoiding Redundancy for Manifold Objects in Boundary Representation

Xiaohong Zhu Beat D. Brüderlin

UUCS-93-018

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

August 16, 1993

Abstract

This paper describes a new approach to the robustness problem in solid modeling. We identify as the main cause of the lack of robustness that interdependent topological relations are derived from approximate data. Disregarding the interdependencies very likely violates basic properties, such as reflexivity, and transitivity, resulting in invalid data representations, such as dangling edges, missing faces, etc. We show that the boundary of manifold objects can be represented without redundant relations which avoids inconsistencies. An algorithm for regularized set operations for manifold solids which is based on the principle of avoiding and eliminating redundancy is described. This algorithm has been implemented for objects bounded by planar and natural quadric surfaces; it handles coincidence and incidence cases between surfaces and curves robustly.

Robust Solid Modeling by Avoiding Redundancy for Manifold Objects in Boundary Representation

Xiaohong Zhu, Beat D. Brüderlin
Department of Computer Science
University of Utah Salt Lake City, UT 84112

Abstract

This paper describes a new approach to the robustness problem in solid modeling. We identify as the main cause of the lack of robustness that interdependent topological relations are derived from approximate data. Disregarding the interdependencies very likely violates basic properties, such as reflexivity, and transitivity, resulting in invalid data representations, such as dangling edges, missing faces, etc. We show that the boundary of manifold objects can be represented without redundant relations which avoids inconsistencies. An algorithm for regularized set operations for manifold solids which is based on the principle of avoiding and eliminating redundancy is described. This algorithm has been implemented for objects bounded by planar and natural quadric surfaces; it handles coincidence and incidence cases between surfaces and curves robustly.

1 Introduction

Geometric objects are defined in a continuous Euclidean space, yet numerical information of representations for representing objects are always discrete (for instance, floating point numbers are used to approximate real numbers). In geometric modeling, most of the construction algorithms involve numerical computations, and compute symbolic information from numerical information. Because of the inaccuracy of numerical data and computation, the construction may fail to construct a new representation that represents a unique geometric object as desired.

Practically, the goal of geometric modeling is to compute representations for desired geometric objects in Euclidean space. If a representation violates any properties of the object it is supposed to represent in Euclidean space, then the representation is *not consistent.* Robustness of a geometric algorithm usually means that the representations computed are *consistent* with the theory of Euclidean space. When geometric information is approximated

by floating-point numbers, floating-point calculations can distinguish object features that are sufficiently apart, but not reliably determine coincidence of objects. In a certain region of proximity, floating-point computation will give seemingly random results because of round-off errors. The problem cannot be addressed satisfactorily by declaring two features coincident whenever they are closer than the tolerance ε. Doing so can easily lead to violations of basic properties, as is demonstrated by the following simple example: When two points P and Q are far enough apart, the decision for $P \neq Q$ is correct and robust. When the points P and Q are closer than twice the tolerance ε, the exact relation between P and Q can not be uniquely determined. If $P = Q$ is chosen, this is an arbitrary decision, and contradictions might occur. For instance, we might have the situation of three points where $P = Q$ and $Q = R$, within the tolerance ε, which implies $P = R$. However, this may contradict the correct tolerance based interpretation $P \neq R$.

Once the relations $P = Q$ and $Q = R$ are decided, the coincidence of P and R is implied by the transitivity. Therefore, explicitly computing relation of the P and R based on numeric data is redundant computation. If data and computation are precise, then this redundantly computed relation is identical to implied information. This is guaranteed by the fact that the numerical representation of coordinates is a model for the Euclidean geometric object. Unfortunately, with approximated numbers the computational error depends on the numerical method applied, yielding different results for the same relation, for instance, when the operations are carried out in a different order or with a different, although symbolically equivalent operation. Many of the representations that are used in geometric modeling contain redundant information in their data structure, and algorithms of construction also involve redundant computations. For this reason, consistency is not guaranteed in geometric modeling, unless infinite precision is applied. It is easy to see from the above example, that we can achieve robustness by either enforcing consistency for redundant operations by explicit reasoning, or by avoiding and removing redundancies. If we already decided that P = Q and Q = R, then we would not have to compare P and R, but would know by inference that they are coincident. Moreover, if two different points are found to be coincident, this represents redundant information. We can replace the representation of point P with the representation of point Q in all occurrences in the data structure. This also helps making comparison with other data consistent. In this paper we develop a solid modeling approach based on avoiding and eliminating redundant information in the data structure.

2 Related Work

A number of researchers have addressed the geometric robustness problem. An overview can be found in [12, 13]. Methods can be classified into the following categories: arbitrary precision, perturbation, symbolic reasoning, tolerance based approaches, and avoiding redundancies.

Some approaches attempt to perform precise computations by using exact numbers to achieve continuity. (such as rational numbers, exact algebraic numbers or a space grid)[10, 22, 31, 32]. Robustness of these algorithms is guaranteed, because no numerical error is introduced. These approaches are based on the assumption that geometric shapes can be represented by exact numbers. This is true only for a limited domain of objects (e.g. polyhedra).

The robustness problem frequently encountered with positional degeneracies, in which cases objects are partially coincident. The methods described in [5, 33] use geometric **perturbation** to avoid coincidence and thus the problem. However, degeneracies are usually intentionally created, such as the construction of assembly parts using the geometric modeler. Therefore, this method excludes many important applications.

Other researchers apply **symbolic reasoning** to guarantee the consistency of interdependent geometric relations[19, 20, 14, 16, 29, 30]. The success of reasoning seems to largely depend on the availability of powerful and efficient symbolic reasoning approaches. In extreme cases, general geometric theorem proving methods[4, 15, 17] have to be employed. However, due to the complexity of the symbolic reasoning problem, these approaches are either mostly limited to relatively simple geometric problems with limited implications by so-called "local" reasoning.

Tolerance-based approaches treat geometric shapes as subsets of a continuum (real numbers), but with limited accuracy. Numeric values are therefore represented as a range, rather than a single value. These approaches have their root in interval arithmetic[21]. A direct geometric generalization of interval arithmetic is "Epsilon Geometry"[11, 26, 28]. In this approach, geometric relations are computed with an error which is propagated by the logical inferences in an algorithm, possibly yielding "undecided" relations. Adaptive tolerance-based approaches have been studied in [2, 3, 6, 7, 8, 9, 28]. These methods adjust the tolerances to keep certain information of the algorithm's decision-making history. When making a new decision, the algorithm will check the tolerances to make the decision consistent

with all the previous ones, and update the tolerances to reflect the new decision. While the results of tolerance-based approaches are generally satisfactory, extra computation is needed for all the tolerance operations. Also these approaches may find ambiguities that have to be fixed by extra means.

Robustness can also be achieved by **avoiding redundancies**. The idea of avoiding redundant computation in set operations has previously been presented in [13]. In [25], a B-rep data structure is found from a CSG representation. Avoiding redundancy in the CSG data structure helps in avoiding redundant decisions in the boundary evaluation. Another, similar approach is taken in [24]. These approaches cannot completely avoid all redundancies, and in general, this is probably too difficult to achieve.

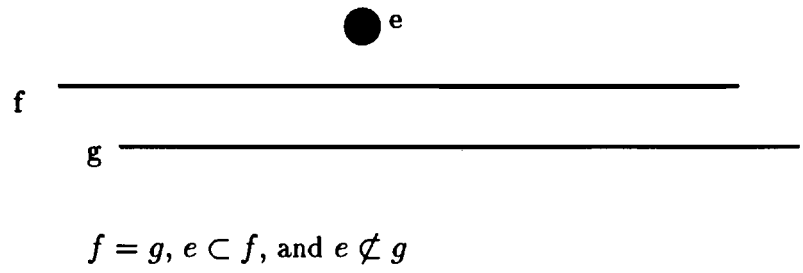
3 Characterization of Redundancy

We observe that the robustness problem is mainly caused by contradictions in redundant and imprecise computations. Before discussing ways of avoiding or eliminating redundancies, we first characterize two types of redundancies: Directly redundant data computation and indirectly redundant data computation.

1. *Directly Redundant Data Computation:* Duplicates of geometric elements may exist in different parts of the data structures of geometric objects. Coincident objects also represent redundant data and it should be obvious that they may cause redundant data computation leading to inconsistencies. For instance (see figure 1), surfaces f and g are coincident within tolerance ($f = g$). When we compare edge e with surface f, we may find that e is on f, but e is not on g within tolerance, which violates the transitivity property of the incidence relation. This *direct redundancy* can be easily avoided by replacing one of the coincident surfaces with the other one. Also, comparing (or intersecting) surface f with g is possibly different from comparing (or intersecting) g with f, violating reflexivity. This problem can be avoided by always comparing the surfaces in the same order.

2. *Indirectly redundant data computation:* Some relations may be implied in different data structures of the representation of the geometric objects.

   For three planes, denoted by f1, f2 and f3 (as shown in figure 2), we first compute the intersection line e of f1 and f2. After this, we compute the relation of line e and the



$f = g$, $e \subset f$, and $e \not\subset g$

Figure 1: Example of direct redundancy

plane f3. If we find line e is on the plane f3, then the following implication applies.

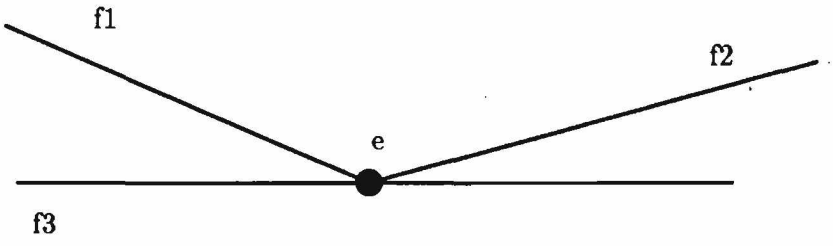
$e = f_1 \cap f_2$ and e on $f_3 \Rightarrow \exists g, h : g = f_1 \cap f_3, h = f_2 \cap f_3, g = h = e$

The intersection of f1 and f3, and the intersection of f2 and f3 are both coincident with e. In other words, the decision that e is on f3, in fact, implicitly computes the intersection of f1 and f3, and f2 and f3. Explicitly computing the other intersections in other parts of an algorithm would therefore create redundant data. The enlargement in figure 2 shows that, due to numerical imprecision there are actually 3 intersection lines. Using a tolerance to decide the relations might determine that e is on f3 (the distance between e and f3 is smaller than the tolerance) but that the three intersections are not coincident (because their relative distance is slightly larger than the tolerance), which is contradictory, according to the above finding.

We call the redundancy caused by such indirectly redundant data computation an *indirect redundancy.*

4 Removing and Avoiding Redundancy

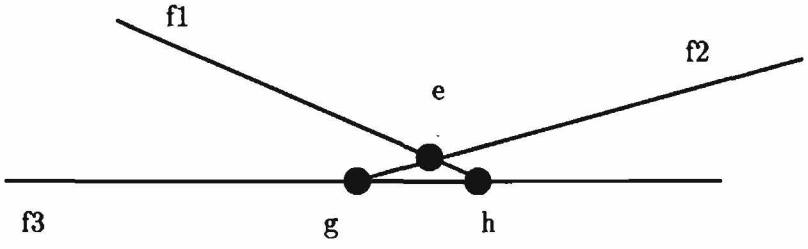
Totally removing redundancies in geometric algorithms may not be realizable in general. In special cases, such as regularized boolean set operation, under the assumption that objects are limited to manifolds, we can attempt to eliminate redundancies. In the following context, we will first explain the definition of manifolds and regularized boolean set operation. After that, we will derive a non-redundant data representation, and operations.



$e = f_1 \cap f_2$ and e is on f_3

$\Rightarrow e = g = h$

where $g = f_2 \cap f_3$ and $h = f_1 \cap f_3$



With tolerance region, e is on f_3, but $g \neq h \neq e$

Figure 2: Three surfaces intersecting in one line

4.1 Manifolds and Non-Redundant Data Representation

Non-manifold solids and mixed topology objects have been introduced for special applications in CAD/CAM. However, many geometric objects that are used in CAD/CAM design, and especially in solid modeling operations, such as Boolean set operations are manifolds. Since robustness in solid modeling is such a difficult issue, it makes sense to aim at a robust solution for the important subclass of manifold objects first. The definitions of manifolds are as follows [12]:

Definition 1 *(3-manifold) A 3-manifold with boundary is homeomorphic to a simplicial complex C of dimension 3 with the following restrictions:*

1. *Every 2 simplex in C is adjacent to one (or two) 3 simplices.*
2. *The link of every 0 simplex is a triangulation of the disk (or the sphere).*

   (the cases in parentheses correspond to interior simplices).

Definition 2 *(2-manifold) A 2-manifold with boundary is homeomorphic to a simplicial complex C of dimension 2 with the following restrictions:*

1. *Every 1 simplex is incident to exactly two 2-simplices.*
2. *The link of every 0 simplex in C is a triangulation of the circle.*

Theorem 1 *M is a connected 3-manifold with boundary; then the boundary of M is an embedded orientable 2 manifold without boundary.*

Theorem 2 *The boundary of a connected 3 manifold is a boundary of a regularized solid in 3D.*

With simplical complices, we here mean *abstract simplical complices* (as defined in[12]) which does not restrict objects to be polyhedral. Some examples for 2-manifold boundaries are illustrated in figure 3. Every edge (1-simplex) is incident to exactly two surfaces. No edge is therefore incident on a third surface, and no two edges are coincident. Also, the link of every vertex (0 simplex) is a triangulation of the circle. Some non-2-manifold boundaries

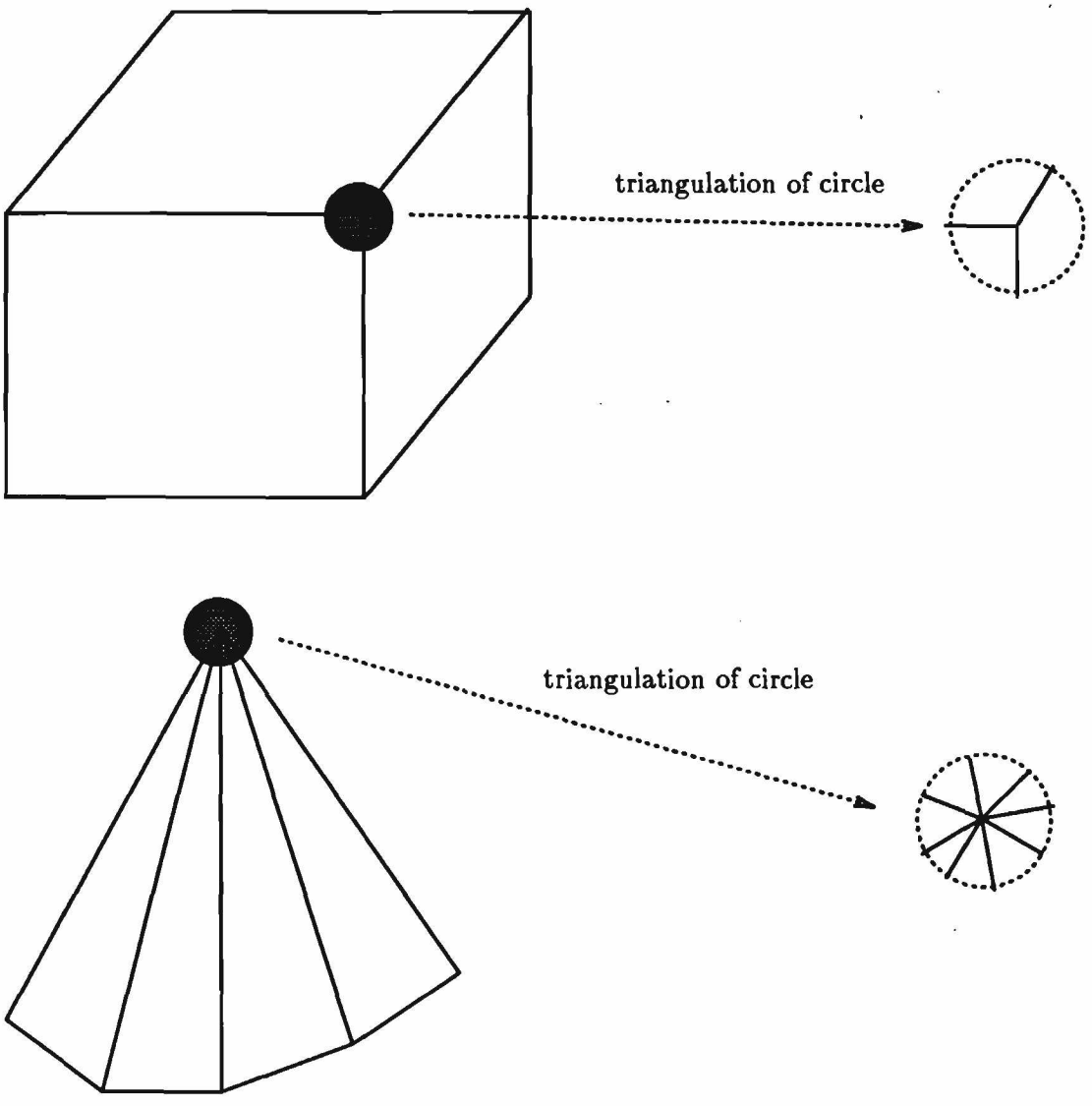


Figure 3: Examples for manifold objects

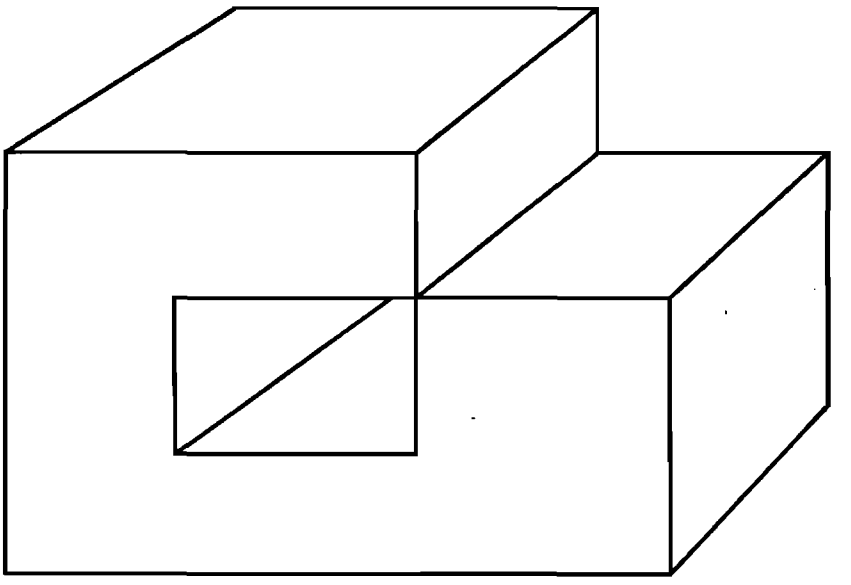


Figure 4: Example for non-manifold

are shown in figures 4 and 5. In figure 4, the object has an edge (1-simplex) incident to four surfaces. In figure 5, there is a vertex (0 simplex) whose link is not a triangulation of the circle (actually it is a triangulation of two circles). In the following we exclude non-manifolds from consideration.

Based on the above definition of manifolds, the boundary representation we use in the set operation algorithm, is slightly different from the well known winged-edge representation described in the literature[1]. A solid object contains a list of bounding faces. Each face is bounded by a set of disjointed edge cycles which we call loops. Each vertex is the intersection of two half-edges that belong to the same face, rather than an intersection of three (or more) faces in 3D. Each edge contains the following topological information :

- 2 incident vertices for each half edge
- the neighbor half edge
- the face it belongs to
- the successor and the predecessor edge in the face the half edge belongs to.
- the direction of the edge (the convention is that the interior of the face is to the left, looking from outside)

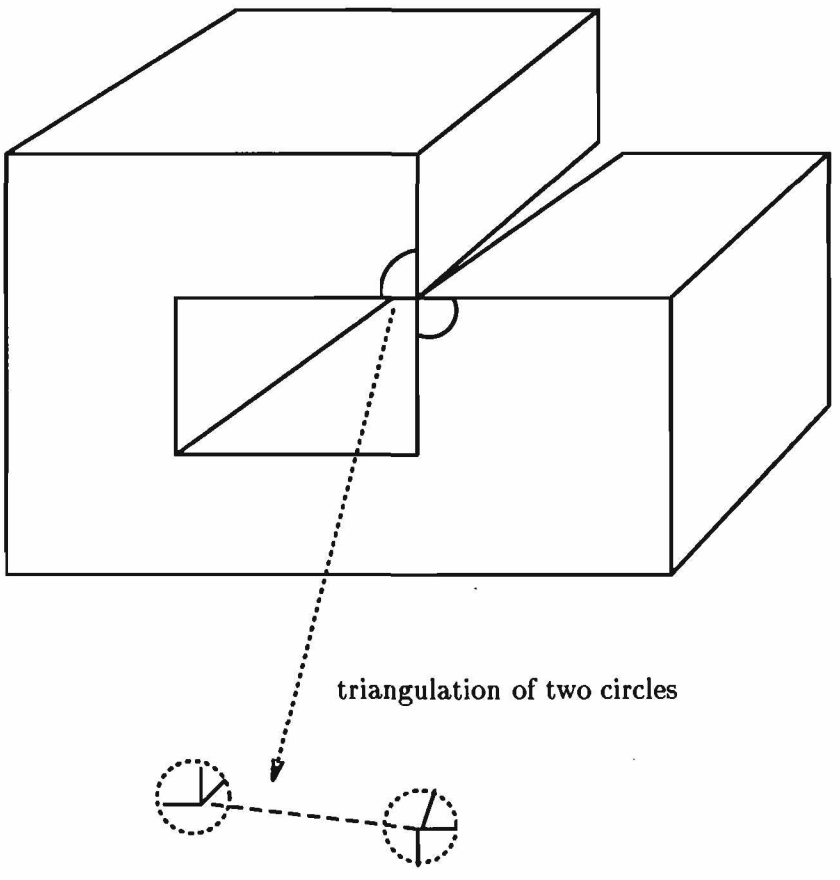


Figure 5: Example for non-manifolds

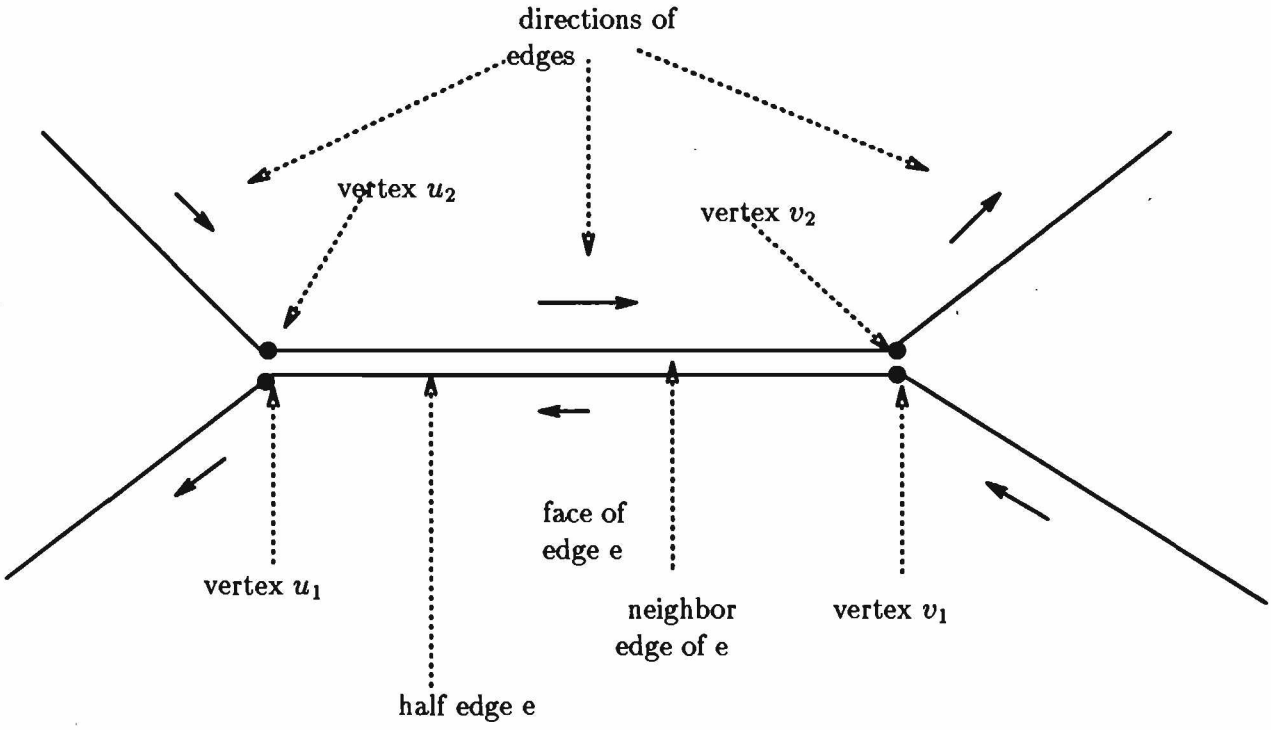


Figure 6: The boundary representation

The edge is oriented by the order of the two incident vertices. One is defined as *start* vertex, and the other is defined as *end* vertex. The schema is shown in figure 6.

4.2 Regularized Boolean Operations

Regularized boolean set operations (*regularized union*, *regularized intersection*, and *regularized difference*) are an important tool used for constructing objects in geometric modeling. They differ from the corresponding set theoretic operations in that the result is the closure of the interior of the solid, which eliminates "dangling" edges and faces, and isolated vertices [23]. Regularized Boolean set operations are used here to handle incident faces and edges uniformly. For a boundary representation, the implementation of regularized Boolean operations is fairly involved. The basic steps are:

- Generate the set membership classifications of the boundaries of one solid with respect to the other solid. This involves computing the intersection between boundaries of different solids and carrying out the inside/outside/on tests for a boundary against a

solid, and the neighborhood evaluation.

- Collect boundaries of the new solid by selecting parts of the boundaries that belong to the resulting objects, depending on the set membership classification, the neighborhood evaluation, and on the type of Boolean operation.
- Build the topological relationships and boundary hierarchy of the new solid.

From the above we can see that the set operation algorithm contains both, "direct" intersection computation (face with face, or edge with edge), and "indirect" intersection computation by computing geometric relations (incidence, coincidence). Both may lead to inconsistency when based on tolerance based relations.

4.3 Avoiding and Removing Redundancies in the Data Representation and Algorithm

In a previous section, we analyzed redundancies occurring in regularized boolean set operations, next we want to either avoid the redundancy, or remove it when it is detected.

For the direct intersection computation, including the face-face, edge-edge intersections, we follow the ideas presented in[14]. When two surfaces are found to be coincident (within the tolerance) we have to replace on of them by the other in the representation of a face. The remaining surfaces are all intersecting (or exactly identical). The surfaces can be ordered (e.g. by their index), and each surface is only intersected with surfaces higher in the order, and thus directly redundant computation can be easily avoided.

For the indirect intersection computation (incidence), we cannot avoid all the redundancies immediately. Fortunately, for manifold objects we can detect redundant data, and eliminate it later based on the type of operation and the neighborhood evaluation.

Relations between edges and faces can be *intersecting, inside, outside*, or *on*.

If an edge is incident on a face, this means that three faces intersect in one common edge. If we can assume that the object resulting from a Boolean set operation is a 2 manifold, then we know that each edge must be the intersection of *exactly* two faces. However, temporarily we may obtain such an *edge-face incidence relation* leading to a redundancy that we need to detect and remove (during the computation of set operations such incidence relations occur relatively often). For example, as shown in figures 7 and 8, e is the intersection edge of face g and face h (Face g, and face h are faces of the same object c1, and face f belongs to object

c2). We determine that e is on the face f. From the topological relations we determine the portions of one of the edges that can be deleted, depending on whether union, intersection, and difference is computed. Therefore, edge e is still an intersection of exactly two faces in the resulting object.

Coincident edges correspond to four faces intersecting in a common intersection curve. Again, in the result we can delete portions of the edges (otherwise we would obtain a non-manifold topology, which we explicitly excluded from the domain of objects we are dealing with here).

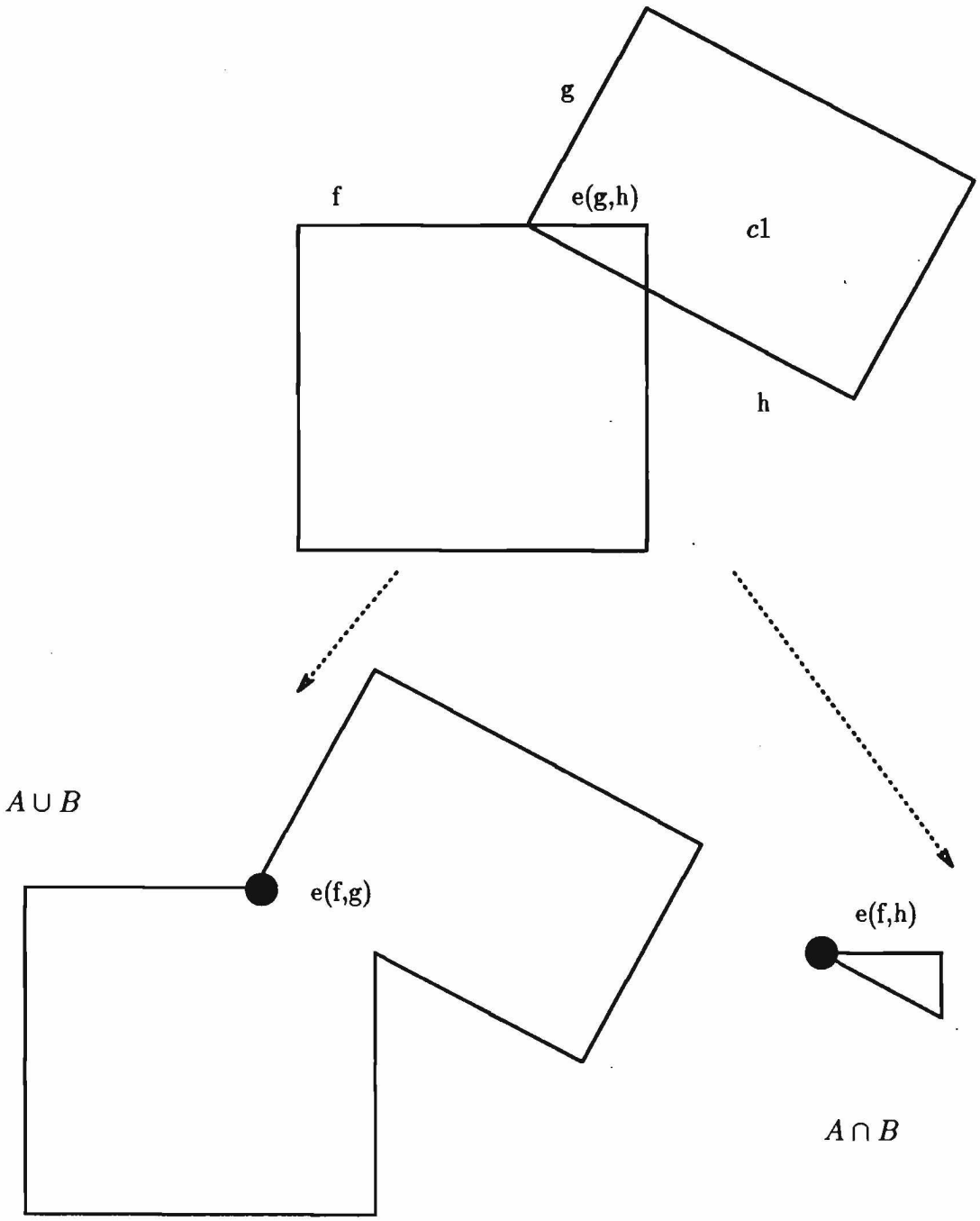
Vertices, in our approach, are computed by intersecting two edges that belong to the same face. Each edge is constructed by intersecting two surfaces, but one of the surfaces is shared by the two intersecting edges, therefore, a vertex is incident on exactly 3 surfaces. In contrast to conventional boundary representation vertices are not shared among faces, here. For instance, for an n-sided pyramid, the top 'vertex' is actually not represented by one, but by n vertices, each is geometrically incident to two edges (i.e. three faces; the face it belongs to, and the two faces adjacent to the two edges). Examples are shown in figures 9 and 10. These vertices are not compared geometrically in 3 space, and therefore not considered coincident, and so redundancy is essentially avoided.

4.4 The Robustness

The reason why most correctly designed geometric algorithms often fail is that incidence and coincidence decisions based approximated data are arbitrary interpretations. When dependencies between such decisions are unrecognized and the decisions are made independently, dependent information may contain contradictions.

In [6] it was shown that relations, such as *apartness* (and thus *inside, outside*) can be determined unambiguously based on a tolerance ε (representing an upper bound on the error), i.e. these relations do not change when the precision is increased (even with infinite precision). This means "apartness" relations are consistent with each other.

On the other hand, relations, such as incidence or coincidence, based on a tolerance are always arbitrary. For instance, if we consider two points that have a relative distance less than ε from each other to be coincident (or apart), this might be inconsistent with other decisions (as explained in the introduction for point coincidence, where the transitivity property of coincidence was violated). Increasing the precision might yield another outcome


g
f
e(g,h)
cl
h
A ∪ B
e(f,g)
e(f,h)
A ∩ B

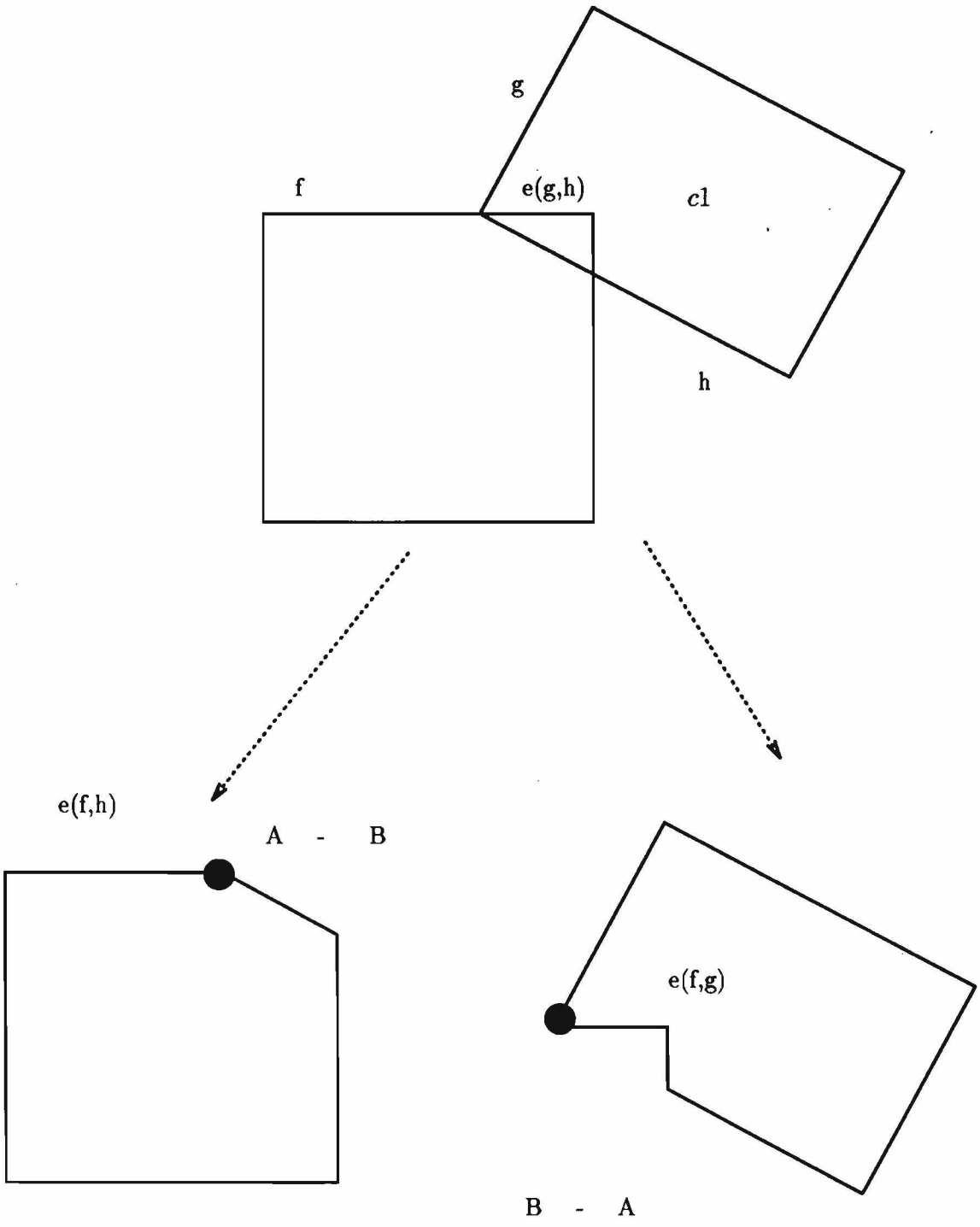


Figure 8: Removing an indirect redundancy for set-differences

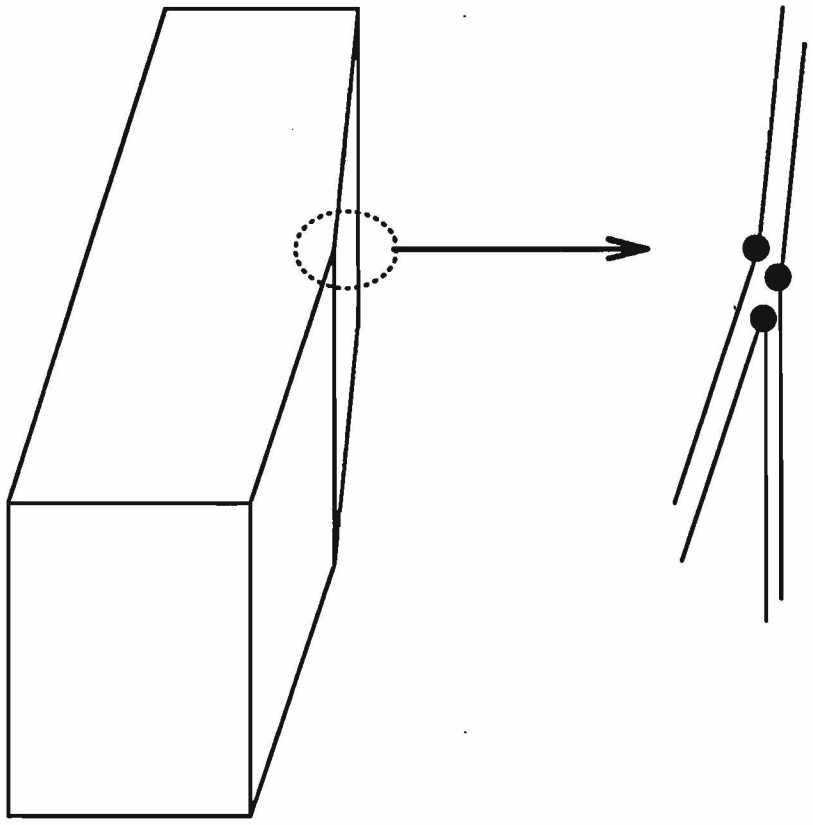


Figure 9: Vertex definition, example 1

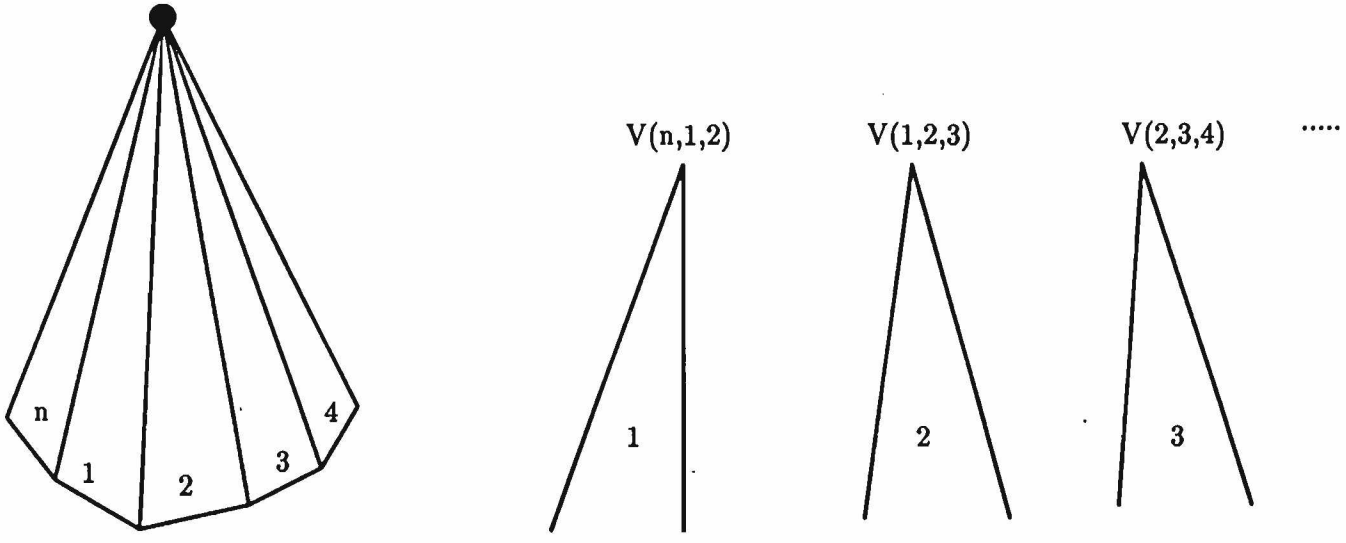


Figure 10: Vertex definition, example 2

of the relation.

If we can avoid such situations altogether, or avoid the dependency between the relations, we can avoid possible inconsistencies. As mentioned above, incidence relations are important in solid modeling, and should therefore be supported. We therefore follow the second path and determine interdependencies of incidence relations and eliminate redundancies in the solid representation, which is possible for the class of 3-manifolds with redundancy. To avoid inconsistencies in the data generated by an algorithm, based on approximated data, we rely on the following lemma:

Lemma 1 *If an algorithm creates consistent and accurate results for consistent and accurate input, on the grounds that all the computations are done exactly (e.g. with infinite precision), the same algorithm also produces consistent results, even with approximated data and operations, if (directly and indirectly) redundant computation is avoided, and redundant data is eliminated before it affects any decision in the algorithm.*

The truth of this lemma is intuitively obvious: Since no decision involving data in the object can be derived from other data or in another way (which is the condition for not being redundant) it therefore cannot contradict any previously made decision.

5 Algorithms

The algorithm for evaluating regularized Boolean expressions presented here, is based on the same basic operations as the algorithms presented in the literature [18, 23] which are intersecting faces, edges, with each other, determining relationships, neighborhood evaluation, and generating a new topology (boundary data structure) from this evaluation Our approach differs in that it takes advantage of the principles of avoiding redundant representations or eliminating them after detection, as described in the previous sections. This is manifested in a different order in which the basic geometric operations are carried out. Also, a slightly modified data structure is used, instead of the winged edge data structure. In [35] we describe a version of this algorithm converting a half space representation into a Boundary representation. The algorithm presented here directly operates on a boundary representation, and does not require the hybrid representation that was used in the previous approach. Nevertheless, all the geometric information is derived from the surface representation (top down, instead of bottom up as it is often done for boundary representations of polyhedra). A

surface geometry representation is associated with each face, indicating which side is inside by the direction of the normal vector (in this sense the face acts as a half space, but no explicit Boolean expression over half spaces are defined for each solid). A curve is defined as the intersection of two surfaces; a point is defined as the intersection of three surfaces. Relations between points, curves and surfaces, are derived from relations between surfaces only.

To eliminate redundant data in the Boolean set operation algorithm we eliminate all coincident surfaces (by merging them). Curves incident on surfaces are limited to intersection curves of 2 surfaces, and vertices are intersections of two edges which is achieved by limiting the solids to 2 manifold topology, where each face has 1-manifold topology. No coincident curves, and curves incident on surfaces, other than the two surfaces that generated the intersection curve will be present in the resulting representation.

The following outline of the set operation algorithm concentrates on the essentials with regards to avoiding redundancies.

Compute Boolean for 2 solids A and B

```
Procedure Boolean(A:solid, B:solid)

Begin

   For each face 'a' of A
      For each 'b' face of B
         Case spatial_relation(a,b) of

         intersecting: find the intersection
           curve(s) and associate them with 'a'
           and 'b' (* as "unlimited" edges e(a,b).
           The edge is represented by 2 half edges;
           one for 'a' and 'b' respectively *)|

         coincident(*within some tolerance*):
           replace 'b' in B with 'a'|
           (* this requires recomputing the boundaries of 'b' *)
         End;
```

```
      od
   od

End Face;
```

After this all the surfaces either intersect or are exactly equal (i.e. they have the same representation).

```
Procedure Edges(A:solid)

Begin

   For each face 'a' of A
      For each intersection edge e(a,b) in 'a'
 compare with all boundary edges e2 of 'a' and 'b'

         (* e(a,b) is the intersection of 'a' and a
         second surface 'b', e2 is the intersection
         of 'a' or 'b' with a third surface 'c', either the
         three surfaces 'a', 'b', and 'c' intersect,
         or e(a,b) and e2 are coincident *)

         first intersect e(a,b) with all boundary edges e2 of
         'a' and 'b' that intersect to determine the range of e(a,b)
         that is inside both 'a' and 'b'.

         second eliminate coincident edges:
            If the range of the two coincident edges overlaps
               the overlapping region is eliminated for
               e(a,b) or e2
         End;
      od
   od
```

End Edges;

The decisions, which part to clip (in case of an intersection), and which overlapping part to eliminate is made based on the neighborhood evaluation and the type of Boolean operations. The principles of neighborhood evaluation and the geometric decisions are described in[23]. The face-face and the edge-edge coincidence relations are computed based on tolerances. More details on this matter can be found in [34]. If the innermost start (end) point of half edge e(a,b) in the above algorithm is incident on the boundary of face 'b', then the corresponding end (start) point of the neighbor half edge in 'a' connects to another intersection edge, and vice versa. To determine which intersection edge to connect to we choose the one with the end (start) point closest to the corresponding start (end) point of the neighbor of e(a,b) (note that the corresponding vertices of two neighbor half edges don't need to be coincident within tolerance). This is unambiguous, as long as there are no more than two edges incident on a single point, in each surface, which the case if the topology of the face's boundary is a one-manifold. In case of a non-manifold face boundary the choice which edge to connect with which is ambiguous. If the neighbor face has a non one-manifold topology as well, the choice can usually be made arbitrarily, but consistency among all the incident edges must be established (e.g by a graph search). Two examples for how the algorithm works are given below.

5.1 Vertex Consistency

CAD applications usually require points to be represented as one vertex in 3D, rather than by point clusters, distributed over multiple faces, as it is done in our approach. With the following definition we can create such a more desirable representation.

If there exists a permutation of indices π and a set of vertices v_j, where j = 1, .., n for $n \geq 3$, and v_j is the intersection of $f_{\pi_{j-1 mod n}} \bigcap f_{\pi_{j mod n}} \bigcap f_{\pi_{j+1 mod n}}$, then the v_j are coincident in 3 space and the links connected to the vertex triangulate a circle.

Practically, this means to connect each vertex in a half edge with the corresponding vertex of the neighbor half edge, then connect that vertex with the corresponding vertex of the neighbor of the other edge intersecting at that vertex, and so on, until we are back to the vertex we started with. Once around, we can associate a unique identifier for a 3D vertex with all the vertices in the cycle. The process terminates once all the vertices in each face point to a corresponding 3D vertex. Note that each decision in the algorithm is uniquely

in face 4	v(2,4,1) becomes v(3,4,1)
	v(3,4,2) becomes v(3,4,1)
in face 2	v(1,2,4) becomes v(1,2,3)
	v(4,2,3) becomes v(1,2,3)

Table 1: Updating Vertex

determined by the winged edge data structure, and no more tolerance based comparisons are required (Actually a comparison of the coordinates might find that the vertices are not coincident within tolerance)

5.2 Examples

The following two examples illustrate the application of the algorithms to different situations.

In figure 11 a), the enlargement of the intersection of surfaces 1, 2, 3, and 4 shows a consistent interpretation of the four surfaces intersecting in 6 vertices distributed over the four faces. Two cycles are created: [v(2,4,1), v(4,1,2), v(1,2,4)] and [v(3,4,2), v(4,2,3), v(2,3,4)], corresponding to two apart points in 3D.

In figure 11 b), the same situation is interpreted with a bigger tolerance in face 4, causing the intersections of the two vertices v(2,4,1) and v(3,4,2),in face 4 to be coincident (within the tolerance), which causes the half-edge h(2,4) between faces 2 and 4 to be deleted in both faces. Consequently the vertices in faces 2 and 4 need to be updated, as shown in table 1. Note that the vertices in faces 1 and 3 are not affected by the update. In the result, only one cycle is created [v(3,4,1), v(4,1,2), v(1,2,3), v(2,3,4)], corresponding to one 3D vertex, where 4 surfaces intersect. The interpretations in 11 (a) and 11 (b) are both topologically consistent.

Another example as shown in figure 12 shows the union of two cylinders with a cube. The intersection is a degenerate case where three surfaces intersect in one curve. The two cylinders intersect in two ellipses (a and a'). One of the ellipses (a) is incident on the top plane of the cube. Therefore the ellipses (b and c) created by intersecting the each cylinder with the plane, are both coincident with the ellipse a. In the resulting object the ellipse a is eliminated, and the ellipses b and c are are split in half (at the vertices where they intersect with a') and only the non overlapping parts of b and c are kept in the result.

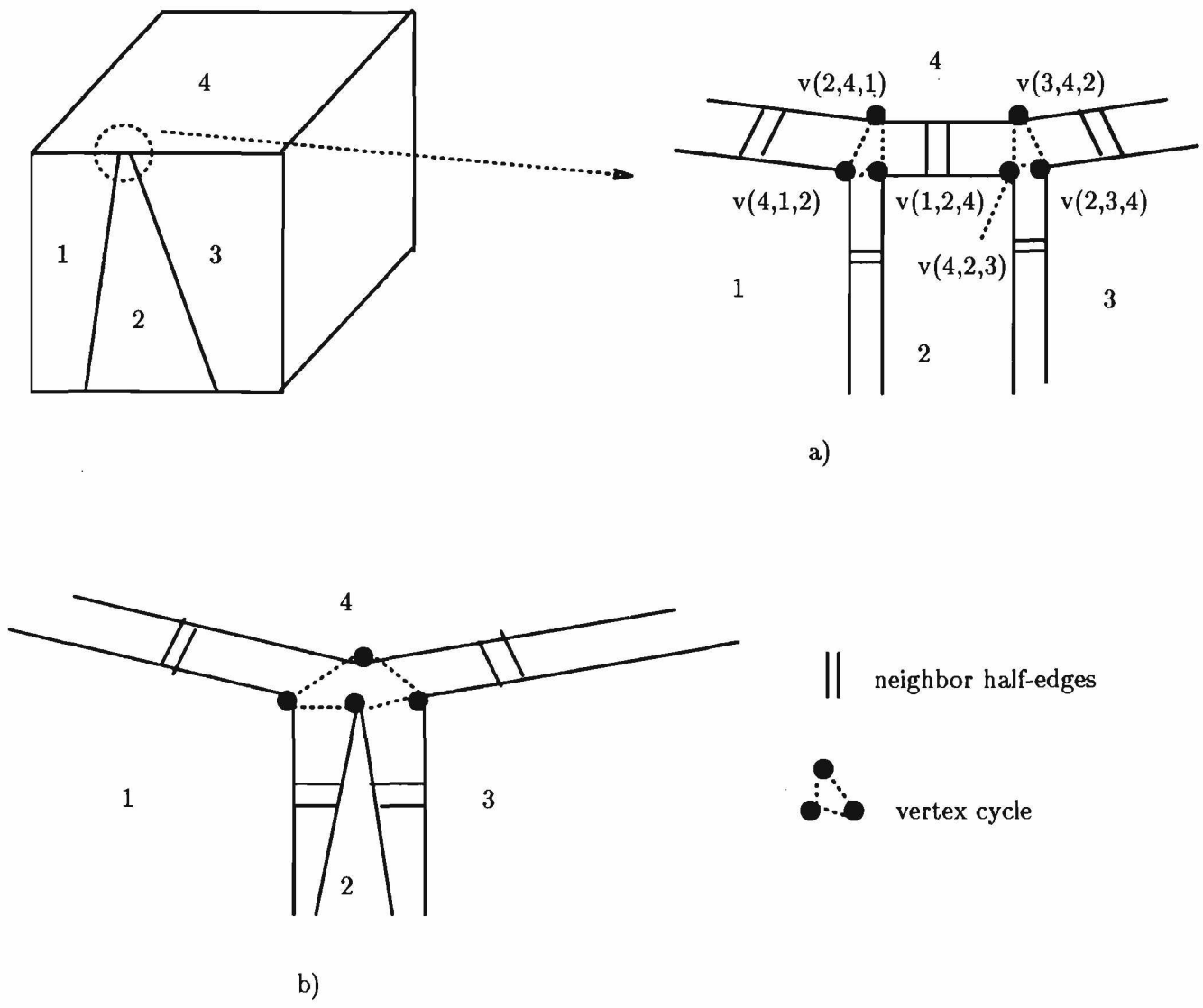


Figure 11: Example for Degenerated Vertices

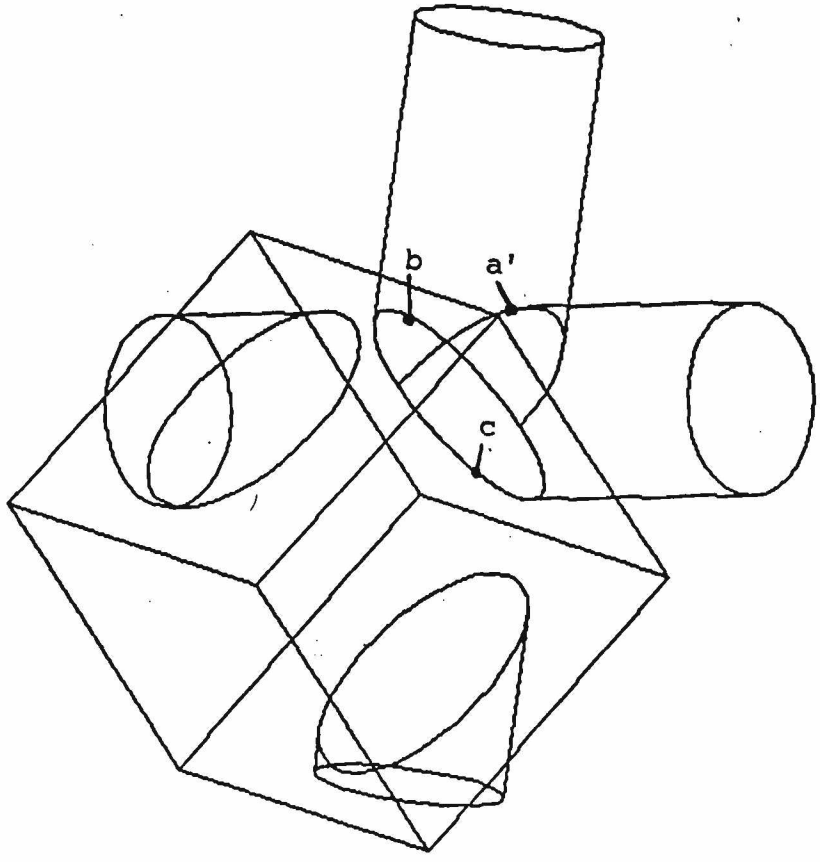


Figure 12: Example for Degenerated Edges

6 Conclusion

The algorithm for Boolean set operations presented in this paper computes the boundary representation of regularized objects bounded by planar and natural quadric surfaces. All special (degenerate) cases are handled properly and robustly despite the fact that floating point arithmetic is used for which incidence decisions have to be made with some tolerance). Robustness is achieved, here, by elimination of redundancy in the data representation which is possible for objects resulting in a 2-manifold topology.

No special considerations needed to be taken to handle the planar and quadric surfaces in the algorithm. The approach would also work with other surface types, such as parametric surfaces, at least in principle. However, the numerical problems of finding intersections and computing incidence relations are definitely more difficult[27].

The approach taken here has the advantage over previous approaches, that no explicit reasoning by the algorithm, or tolerance updates (as in[9]) are necessary. Therefore no ambiguous relations (which would need to be resolved) can occur. This makes the approach simpler and easier to implement, and more efficient at the same time. We believe that the approach can be extended to handle non manifold objects. Non manifold edges can be detected and specially handled by additional means to make their relations with other objects consistent. We could apply the dynamic tolerance approach to handle relations with these non manifold edges (as described in [9]). Local reasoning (as proposed in[29]) could also be applied. Also, perturbation might be applied to make the object a pseudo manifold for the sake of achieving robust Boolean. We can still keep information about the original coincidence of these edges outside the boundary representation (e.g. a list of coincident edges). Simple local reasoning would guarantee transitivity, but not necessarily consistency with other relations in the boundary representation)

7 Acknowledgments

This work has been supported, in part, by NSF grants DDM-89 10229 and ASC-89 20219, and a grant from the Hewlett Packard Laboratories. All opinions, findings, conclusions, or recommendations expressed in this document are those of the authors and do not necessarily reflect the view of the sponsoring agencies.

References

[1] BAUMGART, B. Geometric modeling for computer vision. Ph.D thesis, Comp. Sci. Dept., Stanford University, 1974.

[2] BRUDERLIN, B. Detecting ambiguities: An optimistic approach to robustness problems in computational geometry. Tech. Rep. UUCS 90-003 (submitted), Computer Science Department, University of Utah, April 1990.

[3] BRUDERLIN, B. Robust regularized set operations on polyhedra. In *Proc. of Hawaii International Conference on System Science* (January 1991).

[4] CHOU, S. C. *Mechanical Geometry Theorem Proving.* D. Reidel Publ., Doordrecht, Holland, 1988.

[5] EDELSBRUNNER, H., AND MUCKE, E. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. In *Proc. of 4th ACM Symposium on Comp. Geometry* (June 1988), pp. 118–133.

[6] FANG, S. *Robustness In Geometric Modeling.* PhD thesis, University of Utah, 1992.

[7] FANG, S., AND BRUDERLIN, B. Robustness in geometric modeling – tolerance based methods. In *Computational Geometry – Methods, Algorithms and Applications, International Workshop on Computational Geometry CG'91* (March 1991), Springer Lecture Notes in Computer Science 553, Bern, Switzerland.

[8] FANG, S., AND BRUDERLIN, B. Robust geometric modeling with implicit surfaces. In *Proc. of International Conference on Manufacturing Automation, Hong Kong* (August 1992).

[9] FANG, S., ZHU, X., AND BRUDERLIN, B. Robustness in solid modeling – a tolerance-based, intuitionistic approach. *To appear: Computer-Aided Design (Special Issue on Uncertainties in Geometric Computations* (September 1993).

[10] GREENE, D., AND YAO, F. Finite resolution computational geometry. In *Proc. 27th IEEE Symp. Fundations of Computer Science* (1986), pp. 143–152.

[11] GUIBAS, L., SALESIN, D., AND STOLFI, J. Epsilon geometry: Building robust algorithms from imprecise computations. In *Proc. of 5th ACM Symposium on Computational Geometry* (1989).

[12] HOFFMANN, C. M. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, 1989, ch. 4.

[13] HOFFMANN, C. M. The problems of accuracy and robustness in geometric computation. *IEEE Computer 22*, 3 (March 1989), 31–41.

[14] HOFFMANN, C. M., HOPCROFT, J. E., AND KARASICK, M. S. Robust set operations on polyhedral solids. *IEEE Computer Graphics and Application 9* (November 1989).

[15] KAPUR, D. Using grobner bases to reason about geometry. *J. Symbolic Comp. 2* (1986), 399–408.

[16] KARASICK, M. On the representation and manipulations of rigid solids. Ph.D thesis, McGill University, 1989.

[17] KUTZLER, B. Algebraic approaches to automated geometry proving. Ph.D Diss., Report 88-74.0, Research Institute for Symbolic Comp., Kepler University, Linz, Austria, 1988.

[18] MANTYLA, M. Boolean opeartions of 2-manifolds through vertex neighbourhood classification. *ACM Trans. on Graphics 5*, 1 (January 1986), 45–60.

[19] MILENKOVIC, V. Verifiable implementations of geometric algorithm using finite precision arithmetic. *Artificial Intelligence 37* (1988), 377–401.

[20] MILENKOVIC, V. Verifiable implementations of geometric algorithm using finite precision arithmetic. Ph.D thesis, Carnegie Mellon University, 1988.

[21] MOORE, R. E. *Interval Analysis.* Prentice-Hall, 1966.

[22] OTTMANN, T., THIEMT, G., AND ULLRICH, C. Numerical stability of geometric algorithms. In *ACM Annual Symposium on Computational Geometry* (June 1987), pp. 119–125.

[23] REQUICHA, A. A. G., AND VOELCKER, H. B. Boolean opeartions in solid modeling: Boundary evaluation and merging algorithms. *Proc. IEEE* (1985), 30–44.

[24] ROSSIGNAC, J. R., AND O'CONNOR, M. A dimension-independent model for pointsets with internal structures and incomplete boundaries. *Geometric Modeling for Product Engineering* (September 1989), 145–180.

[25] ROSSIGNAC, J. R., AND VOELCKER, H. B. Active zones in csg for accelerating boundary evaluation, redundancy elimination, interference detection and shading algorithms. *ACM Transactions on Graphics* (January 1989), 51–87.

[26] SALESIN, D., STOLFI, J., AND GUIBAS, L. Epsilon geometry: Building robust algorithms from imprecise calculations. In *ACM Annual Symposium on Computational Geometry* (1989), pp. 208–217.

[27] SEDERBERG, T. W. Implicit and parametric curves and surfaces for computer aided geometric design. Ph.D thesis, Mech. Engr., Purdue University, 1983.

[28] SEGAL, M. Using tolerances to guarantee valid polyhedral modeling results. *Computer Graphics 24*, 4 (1990), 105–114.

[29] STEWART, A. J. Robust point location in approximate polygons. In *1991 Canadian Conference on Computational Geometry* (August 1991), pp. 179–182.

[30] STEWART, A. J. The theory and practice of robust geometric computation, or, how to build robust solid modelers. Ph.D Thesis 91-1229, Department of Computer Science, Cornell University, 1991.

[31] SUGIHARA, K., AND IRI, M. Geometric algorithms in finite precision arithmetic. Res. Mem. 88-14, Math. Eng. and Information Physicas, University of Tokyo, 1988.

[32] SUGIHARA, K., AND IRI, M. A solid modeling system free from topological inconsistency. *Journal of Information Processing 12*, 4 (1989), 380–393.

[33] YAP, C. K. A geometric consistency theorem for a symbolic perturbation theorem. In *Proc. of 4th ACM Symposium on Comp. Geometry* (June 1988), pp. 134–142.

[34] ZHU, X. Consistent geometric modeling approaches. *Master Thesis* (1993).

[35] ZHU, X., FANG, S., AND BRUDERLIN, B. Obtaining robust boolean set operation for manifold solids by avoiding and eliminating redundancy. In *Proc. of Second Symposium on Solid Modeling and Applications* (May 1993).