Robustness in Geometric Modeling – An Intuitionistic and Tolerance-based Approach.

Shiaofen Fang, Beat D. Bruderlin

UUCS-92-046

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

December 23, 1992

Abstract

An intuitionistic geometry approach is taken to develop two tolerance-based methods for robust geometric computation. The so called analytic model method and the approximated model method are developed independently of a specific application or a geometric algorithm. Geometric robustness is formally defined. Geometric relations are computed based on tolerances defined for geometric objects. Dynamic tolerance updating rules are given to preserve properties of the geometric relations. The two methods differ in the definition of robustness and they use different tolerance updating rules, and hence, they preseve different properties and are suitable for different kinds of applications. To handle the possibly occuring ambiguities dynamic ambiguity handling methods are described as well.

Robustness in Geometric Computation – An Intuitionistic and Tolerance-based Approach

Shiaofen Fang   Beat Brüderlin
Computer Science Department
University of Utah
Salt Lake City, UT 84112

Abstract: An intuitionistic geometry approach is taken to develop two tolerance-based methods for robust geometric computation. The so called analytic model method and the approximated model method are developed independently of a specific application or a geometric algorithm. Geometric robustness is formally defined. Geometric relations are computed based on tolerances defined for geometric objects. Dynamic tolerance updating rules are given to preserve properties of the geometric relations. The two methods differ in the definition of robustness and they use different tolerance updating rules, and hence, they preseve different properties and are suitable for different kinds of applications. To handle the possibly occuring ambiguities dynamic ambiguity handling methods are described as well.

1 Introduction

1.1 Robustness Problem in Geometric Computation

Geometric computation plays an important role in applications such as CAD/CAM, robotics, computer graphics and computer animation. Nevertheless, it seems that an important obstacle in the way of successful large scale applications of geometric computation is its lack of robustness. Practical implementations of geometric operations remain error-prone[1]. This problem becomes even more evident in geometric computation with higher degree curves and surfaces (e.g., in sculptured solid modeling).

Floating point arithmetic errors and approximation errors are the two kinds of errors most commonly encountered in geometric computation. The floating point arithmetic used in computers to simulate real numbers has only finite precision for the representation and computation of real numbers, whereas geometric algorithms are often based on a specification with perfect real number arithmetic. The cumulative effect of such errors can lead to disastrous results in the representation and computation of geometric shapes and relations, especially in situations that are sensitive to small errors such as degenerate geometric relations (e.g., coincident lines or tangent surfaces). Also, a closed form computation for higher degree curves and surfaces (e.g., NURB curves and surfaces) is usually impractical or impossible, and approximations to these curves and surfaces with polygons and polyhedra are often necessary. These approximations inevitably will create errors. Numerical iteration methods, which are often used to find an approximated solution for an intersection, also generate errors.

With these errors, geometric relations cannot be determined with absolute certainty. Tolerances are often introduced to reflect these errors especially to determine degenerate cases which are very common and mostly intentional in many geometric problems. For instance, two points are considered coincident if their Euclidean distance is smaller than the given tolerance. A line segment is incident on a plane if both end points have a distance from the plane, smaller than the tolerance. Two planes are coincident or parallel, if the two surface normals are at an angle smaller than some tolerance, etc. Unfortunately, decisions based on this simple use of tolerances are arbitrary and can cause inconsistencies. An example is given in Figure 1 in which the union of two cubes is computed. Edge AB is first determined to be intersecting with plane P using the tolerance criterion for line-plane incidence. Later planes P and Q are compared and detected to be coplanar, using the tolerance criterion for two planes. However, the two decisions are not independent – since the line is incident on plane Q (it is a boundary line of plane Q in the cube), it should also be incident on all the planes that are coincident with Q. The computation based on such inconsistent relations result in the invalid solid representation of Figure 1(b). More such examples are discussed in [10].

The correctness of an algorithm relies on the consistency of the relations computed during its execution. Because these inconsistencies are not usually expected and explicitly tested for, the algorithm may fail abruptly and without apparent reason, or invalid representations are created as a result.

1.2 Related Work

A survey on robustness in geometric computation has been given in [11]. A number of publications also addressed this problem recently.

Precise computations are performed by using only exact numbers (e.g., bounded or un-

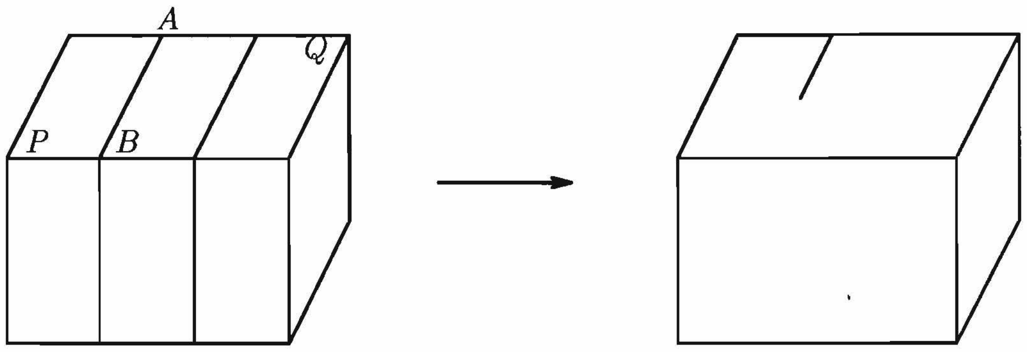


Figure 1: The union of two cubes. (a) The two original cubes; (b) The invalid union (yielding a dangling edge)

bounded rational numbers, exact algebraic numbers or space grids) [8], [22], [26] and [27]. The approaches are based on the supposition that geometric shapes can be represented using exact numbers and all computation on them should only be performed using exact numbers. Currently, it is very inefficient to use precise computation on exact numbers when only floating point arithmetic is efficiently supported in modern computers. Also, because not all geometric features can be represented with rational numbers, geometric and even topological changes (e.g., vertex shifting and edge cracking) are often necessary[18][19], which drastically limits the domain of representable objects.

In [5] and [29], an approach is taken where the input data are perturbed by a small amount to avoid positional degeneracy. This approach obviously cannot represent intentionally degenerate cases occuring often in computer aided design, and is therefore unsatisfactory for many applications.

Symbolic reasoning has been used in some approaches to maintain the consistency among all the decisions made regarding geometric relations. However, to strictly adhere to all the symbolic constraints necessitates a theorem proving process[4][14][16] which is usually too complicated for practical applications. In general, the symbolic reasoning approach is limited to simple geometric problems, or applied locally to specific geometric features. Hoffmann, Hopcroft and Karasick[13][15] presented a polyhedral intersection algorithm using a few simple robustness rules. Milenkovic[18][19][20] developed an algorithm computing the arrangement of a set of pseudolines with an input set of lines. The pseudolines are implicitly defined curves by a set of rules that guarantees that the pseudolines are within a certain distance to their original lines. Stewart[24][25] proposed a concept of local robustness for polyhedral intersection algorithm. It uses local symbolic reasoning with traditional constraint propagation to get a consistent set of relations within the local features.

Tolerance-based approaches keep sufficient information about uncertainty regions (tolerances) of geometric objects and update the tolerances according to the decisions made in order to maintain the consistency of the decisions[2][3][6][7][23]. The ideas of this approach are partially related to interval arithmetic[21]. A geometric instance of interval arithmetic is "Epsilon Geometry"[9], which provides a logical framework for algorithms that compute a consistent solution for a perturbed version of the input and return a bound on the size of the implicit perturbation. An explicit use of dynamic tolerances can be found in a polyhedral Boolean operation algorithm developed by Segal[23]. In this algorithm, geometric relations are detected based on the tolerances of geometric objects. Tolerances grow upon the detections of geometric relations. The changes of the tolerances are propagated to related features by means of symbolic reasoning mechanism. No proof of robustness is given, however. In [2] and [3] Bruderlin presented a similar approach with more rigorous robustness rules for tolerance updating. It is proved that if no ambiguity is found the algorithm is robust in the sense that there exists a model which corresponds to the decisions of the algorithm.

1.3 An Overview

This paper presents two methods for robustness in geometric computation. The approach taken is intuitionistic and tolerance-based. It constructs geometric relations using tolerances defined for the geometric objects and dynamically updates the tolerances to preserve the properties of the geometric relations. The two methods, the *analytic model method* and the *approximated model method*, are developed from two different definitions of geometric robustness respectively. The methods are essentially independent of the algorithms to which they are applied.

In section 2, geometric robustness is formally defined. The two general methods, the *analytic model method* and the *approximated model method*, are presented in section 3 and section 4, respectively. Ambiguity problems are discussed in section 5. An ambiguity handling approach is given there to try to dynamically adjust tolerances to solve ambiguities.

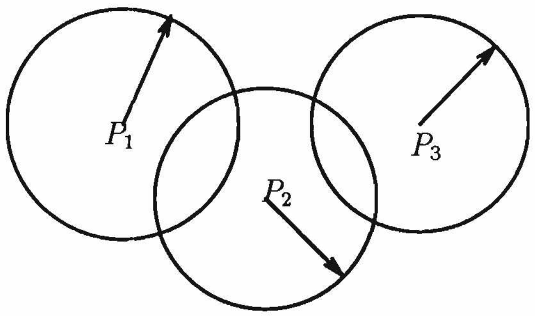


Figure 2: Comparing three points

2 Definitions of Robustness

2.1 The Inconsistency Problem

A common practice in geometric computation to detect degenerate cases with inaccurate data is to use tolerances. Usually, the tolerance is defined to be the error τ accumulated during previous operations. When the distance of two points, for example, is less than 2τ, they are considered close enough and decided to be coincident, otherwise they are considered apart. However a closer look at this approach reveals that this definition of the coincidence relation is problematic. For instance in Figure 2, it is first found that P_1 and P_3 are apart ($P_1 \neq P_3$), and P_1 and P_2 are coincident ($P_1 = P_2$), then P_2 and P_3 are found coincident ($P_2 = P_3$). $P_1 = P_2$ and $P_2 = P_3$ lead to $P_1 = P_3$ (by means of the transitivity of coincidence) which contradicts an earlier decision of $P_1 \neq P_3$ Thus, the coincidence and apartness relations are not consistent.

The reason for the inconsistency problem is the following: When two points with inexact coordinates are close within the given tolerance, deciding they are coincident or apart is arbitrary. Moreover, these geometric relations are supposed to have intrinsic properties such as the theorems of Euclidean geometry, which are likely to be violated by these arbitrary decisions. For instance, in the point coincidence/apartness test example, the coincidence relation does not have a transitivity property, which is usually relied upon in the correctness analysis of a geometric algorithm.

2.2 Intuitionistic Geometry

To solve inconsistency problems such as the one in last section, the concept of intuitionistic geometry will be introduced in this section and will be used throughout the following sections.

The term "intuitionistic geometry" is derived from so-called intuitionistic mathematics or constructive mathematics[28]. Intuitionists believe that the subject matter of mathematics consists of the independent existence of mathematical objects; therefore non-constructive inferences and propositions should be rejected. In particular, the law of the excluded middle $\varphi \bigvee \neg\varphi$, which is an axiom of classical logic, cannot be used in intuitionistic mathematics. To prove for some predicate φ that $\varphi \bigvee \neg\varphi$ holds, in intuitionistic mathematics one has to show either that the construction corresponding to φ can be effected or that the construction corresponding to the refutation of φ can be effected. An example for this is the equality of real numbers. In [28] it is shown that the equality relation of real numbers is constructively undecidable. In other words, for two real numbers we may find that they are unequal after finite computation but we can not conclusively decide that they are equal. The dilemma is dealt with by introducing a new relation $\neq$ (apartness). For two real numbers a and b, $\neg(a = b) \not\equiv (a \neq b)$ (unlike in classical algebra). The fact that $(a = b) \bigvee (a \neq b)$ does not hold can be interpreted by introducing a third relation "ambiguity" where we cannot decide whether two numbers are equal or apart.

The Euclidean space (defined over the real number field) is a model for Euclidean geometry which underlies most geometric algorithms in geometric modeling. Implementations of these algorithms have to compute (in other words, construct) coordinates of geometric objects and to compute relations between objects such as coincidence or incidence. Because Euclidean geometry is defined over real coordinates and equality for real numbers is undecidable, it is obvious that incidence relations also are undecidable. This, in turn, lets us conclude that an intuitionistic approach more accurately reflects the behavior of geometric algorithms based on floating point numbers than classical logic. Failure to recognize this seems to be the main reason for the unexpected failure of algorithms in many current applications.

In this paper, we pursue the intuitionistic approach to relate those ideas to geometry in the following ways.

- To explain why geometric algorithms based on real numbers fail in general unless the algorithm does some problem specific specific consistency tests (which are usually too complicated in practice).

- To constructively identify those cases where an unambiguous and consistent interpretation of geometric relations based on tolerances is possible.
- Find constructive definitions of such relations as incidence, apartness, and ambiguity.

2.3 Representation and Model

Before formally defining robustness and actually constructing geometric relations using tolerances, the concepts of representation and model is introduced as a useful tool. The notion of representation and model for general geometric problems was introduced in [12]. A geometric object is represented in a computer by some representation.

Definition 1 *(representation)*

A representation is a data structure intended to describe a geometric object in Euclidean space. It consists of three parts.

- *Symbolic form: Specifying the mathematical form (syntax) of the representation.*
- *Numerical data: Contain numeric values for the parameters in the symbolic form.*
- *Constraints: Describe the relationships with other geometric objects.*

□

Definition 2 *(model)*

A representation has a model if there exists a geometric object in Euclidean space satisfying all the constraints and symbolic forms of this representation. □

When working with inaccurate data and computations the numerical data of a representation cannot not generally satisfy all the constraints. As long as there exists some geometric object in the uncertainty of the values in Euclidean space satisfying these constraints and symbolic forms, the representation is meaningful.

For example, according to the pascal theorem, nine lines can have the arrangement with nine degenerate intersection points (three lines intersect in one point) as shown in Figure 3. Because of the computational and representational errors, the numerical data of the computer representation of these nine lines may violate the pascal theorem as it is shown in Figure 4.

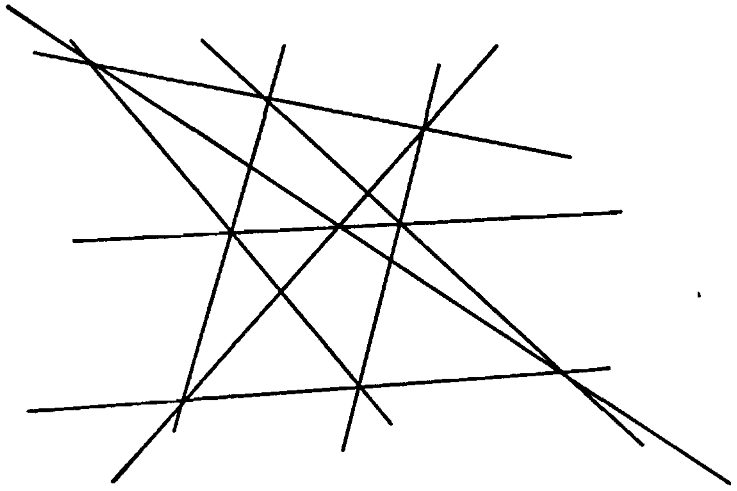


Figure 3: Representation and model example: pascal theorem

There is no model within the tolerance regions of the lines and points, that would satisfy the relations. Obviously, the existence of a model should be an important factor in robustness. The following robustness definition is given in [11] and [10].

Definition 3 *(robustness of an algorithm)*

Assume $op(x_1, x_2, ..., x_k)$ *is a k-ary geometric operation, where* x_i *is a variable in domain* $\mathcal{D}_i$*, and* $\mathcal{A}(R_1, R_2, ..., R_k)$ *is an algorithm implementing operation op, where* R_i *is the representation of variable* x_i*.* $\mathcal{A}$ *is robust iff if for every legitimate input configuration, represented by* R_i *there exists a model* M_i *such that the following is true:*

- *the algorithm constructs an output representation* $R = \mathcal{A}(R_1, R_2, ..., R_k)$ *without failing, and*
- *there is a model* M *of* R *such that* $M = op(M_1, M_2, ..., M_k)$*.* $\square$

2.4 Consistency

A related concept of robustness is that the relations derived in the algorithm should be consistent[2]. Here the consistency usually refers to the fact that the relations detected or defined in the algorithm should not contradict each other with respect to some theory T.

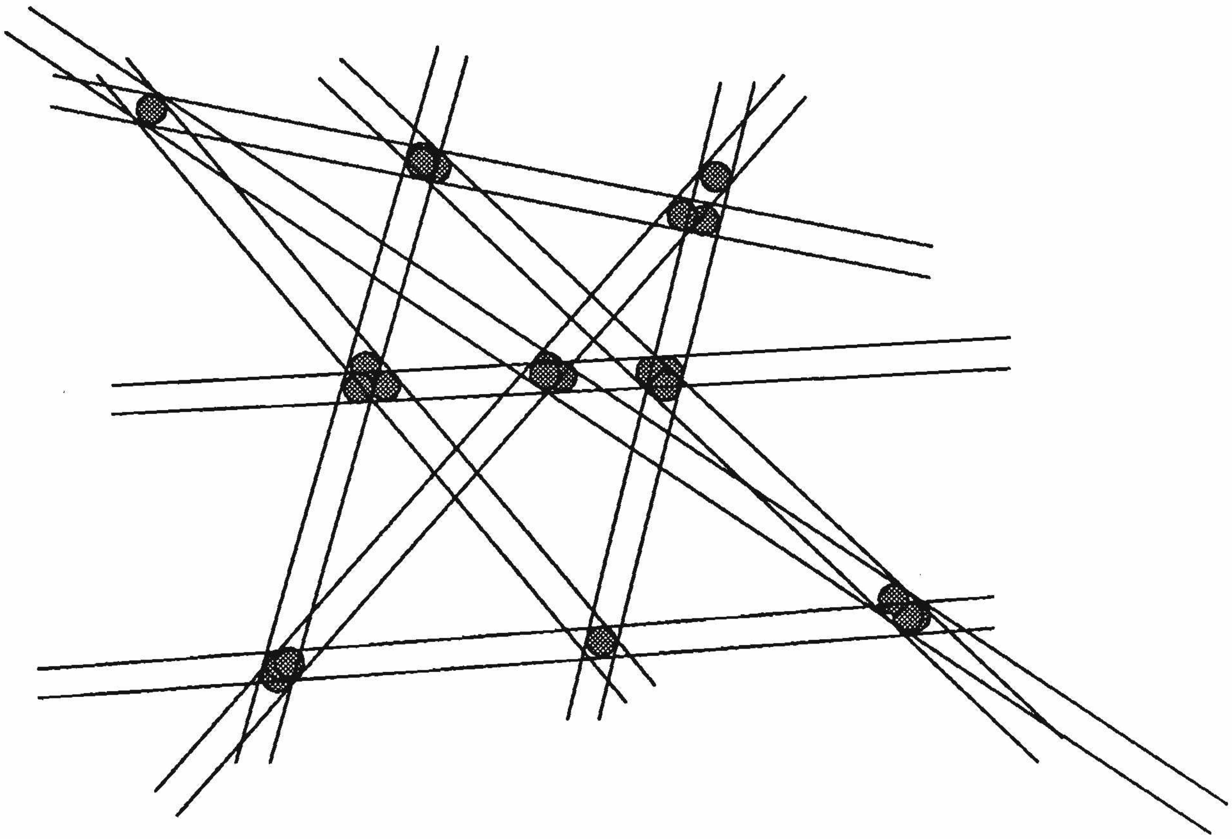


Figure 4: An inconsistent configuration of the pascal theorem using a simple tolerance based incidence definition

The result in Figure 1 is inconsistent because the intersecting relation of edge AB and plane P contradicts the relations that AB is on plane Q and Q is coincident with P which is inconsistent with the theory of Euclidean geometry.

Definition 4 *(consistency)*

A representation is consistent if it has a model. □

The next theorem claims that robustness can be achieved by keeping all the constraints consistent (by definition, there is a model satisfying all these constraints).

Theorem 1 *Assume $op(x_1, x_2, ..., x_k)$ is a k-ary geometric operation as in definition 3, and $\mathcal{A}(R_1, R_2, ..., R_k)$ is an algorithm implementing operation op, where R_i is the representation of variable x_i. The implementation of the operation op by $\mathcal{A}$ is robust according to definition 3, if algorithm $\mathcal{A}$ is correct in Euclidean space (i.e., for every legitimate input M_i in Euclidean space, $\mathcal{A}'(M_1, M_2, ..., M_k) = op(M_1, M_2, ..., M_k)$, where $\mathcal{A}'$ is an algorithm identical to $\mathcal{A}$ except that all the primitive operations of $\mathcal{A}'$ are operations in Euclidean space with precise computation), and if the representation for all the objects in both the input and output and for the objects created during the execution of the algorithm $\mathcal{A}$ is consistent (definition 4).*

Proof: From the definition of consistency, there exists a model satisfying all the constraints and symbolic forms defined in both the input and the output representations and created during the execution of the algorithm $\mathcal{A}$. Obviously, this model is also a model of all the input and the output representations.

Also, because the representation for all the objects in both the input and output and for the objects created during the execution of the algorithm $\mathcal{A}$ is consistent, each step of the algorithm on representations has an equivalent step in Euclidean space on the models of these representations. In other words, the execution of the algorithm $\mathcal{A}$ on representations is equivalent to its execution on the models of these representations in Euclidean space. Therefore the model of the output representation M will be the result of the ideal execution of the algorithm with the models of input representations $M_1, M_2, ..., M_k$ as inputs ($M = op(M_1, M_2, ..., M_k)$). According to definition 3, the implementation of op is robust. □

Definition 3 requires that the model of the output representation M satisfy condition $M = op(M_1, M_2, ..., M_k)$, where $M_1, M_2, ..., M_k$ are the models of the input representations. Unfortunately, this condition can only be tested in an ideal computational environment (i.e., Euclidean space), but not directly with finite precision computation. Theorem 1 provides a practical means for achieving robustness, namely keeping the consistency of the representation of not only the input and output objects but also the objects created during the execution of the algorithm. In other words, as long as the representation of all the objects (and relations) computed during the execution of the algorithm is consistent (i.e., there exists a model), and as long as the algorithm is logically correct, the conditions in definition 3 are automatically satisfied.

2.5 Tolerance Restriction, Analytic Model and Approximated Model

Definition 3 does not specify the relationship between the model and the numerical data of the representation. However, in practice, it is often required that the model is in the uncertainty region of the numerical data. A natural way of keeping the model close to the numerical data of the representation is to use tolerances, and then require models to satisfy certain tolerance restrictions.

Next, the basic concepts of simple and complex objects, the finite part of a simple object, and region will be introduced. They will be used in the following context.

Definition 5 *(simple object, complex object)*

- *A simple object is the representation of a k-manifold in E^3, where $k < 3$, i.e., it is either a point (0-manifold), a curve (1-manifold) or a surface (2-manifold).*
- *A complex object O consists of a set of simple objects related (directly or indirectly) by incidence and coincidence relations, i.e., $\forall$ pairs of objects O_1 and O_n in the complex object, $\exists$ a sequence of simple objects (including O_1 and O_n) $O_1, O_2, ..., O_n$ in O, where $n \geq 1$ and is finite, so that O_i is incident on O_{i+1} or O_{i+1} is incident on O_i, or O_i and O_{i+1} are coincident ($i = 1, ..., n-1$).* □

An example of a complex object is the set of all the simple objects in the representation of a cube. It consists of 6 planes, 12 lines, 8 points and a set of incidence relations connecting them. Each point is incident on three lines. Each line is incident on two planes. Relations, such as inside or outside are not part of our representation of complex objects. The lines and planes of the representation of a cube therefore extend beyond the range of the actual cube and are limited only by the finite universe U (see definition 7.)

Definition 6 *(representative)*

The representative of a simple object is a geometric object (a model) in Euclidean space that is exactly defined by the numerical data and symbolic form of the representation of this simple object. No relations with other simple objects are defined for the representative.

Definition 7 *(universe, finite part)*

- *The universe U in E^3 is defined as:*

$$U(C, R) = \{p = (x, y, z) \mid d(C, p) < R\}$$

  where $d(C,p)$ is the Euclidean distance of points C and p, C is a fixed center point and R is a large enough radius so that all geometric operations of the algorithm are performed inside U.

- *The finite part of a geometric object in Euclidean space is the part of this object that is inside the universe. The finite part of a simple object is the finite part of the representative of this simple object.* □

Now assume that an initial tolerance τ is given ($\tau > 0$). The tolerance restriction with τ is defined as follows.

Definition 8 *(region, tolerance restriction)*

- *An* r-region *of a simple object O is a subset of the Universe U in which each point has an Euclidean distance less than r from the representative of O, where r is a real number, called the* radius *of this region.*

- *A set of geometric objects M in Euclidean space satisfies the tolerance restriction of an object O for a given tolerance τ if there exists a one-to-one mapping ϕ between each object in M and the representative R of O, $M = \phi(O)$, such that:*

  - *The finite part of each object in M is inside the τ region of the representative R, such that $\forall\ v \in R$, $d(u,\phi\ (u)) < \tau$.*
  - $\forall\ u \in M,\ v \in R : u = \phi(v)\ \nexists\ w$ *in M: $(u\text{-}v) = \lambda\ (w\text{-}v)$ for real numbers λ.* □

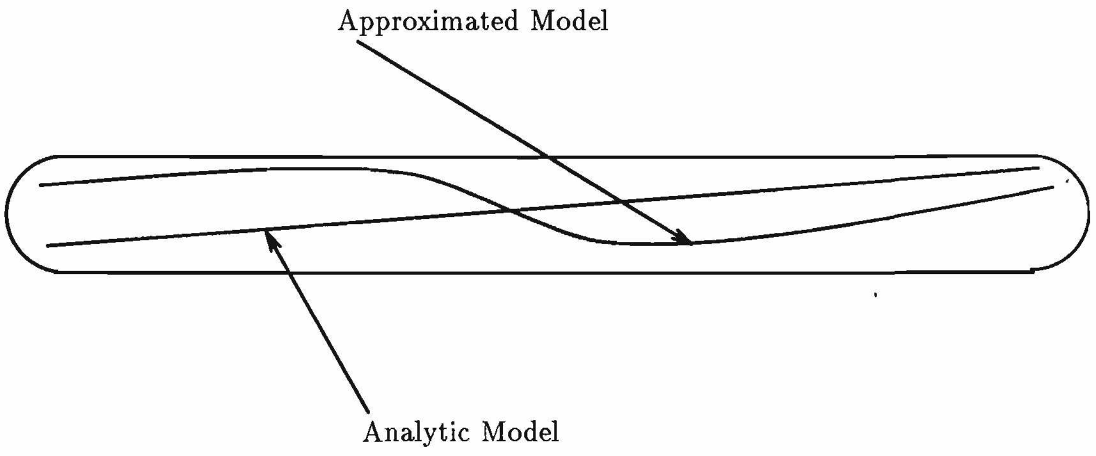


Figure 5: Example of the analytic model and approximated model of a line representation

Definition 9 *(analytic model, approximated model)*

Given a tolerance τ.

- *An object M (consisting of a set of simple objects) in Euclidean space is an analytic model of a representation R iff M satisfies the tolerance restriction of R and the simple objects in M have the same symbolic form as R and satisfy all the constraints requested for R.*
- *An object M (consisting of a set of simple objects) in Euclidean space is an approximated model of a representation R (consisting of a set of simple object representations and their relations) iff M satisfies the tolerance restriction of R and the simple objects in M satisfy all the constraints of R, but they may have a different symbolic form than R.* □

An example of the analytic model and approximated model for a representation of a line is shown in Figure 5.

2.6 Tolerance-Based Robustness

With the definitions of analytic model and approximated model, the consistency definition can be rewritten as follows.

Definition 10 *(analytically consistent, approximately consistent)*

A representation is analytically consistent if it has an analytic model.

A representation is approximately consistent if it has an approximated model. □

Combining theorem 1 and definition 10, tolerance-based robustness will be defined and serve as the sole standard robustness definition in the rest of this paper.

Definition 11 *(tolerance-based robustness)*

A geometric algorithm is analytically (approximately) robust if for every legitimate input representation

- *The algorithm constructs an output representation R without failing.*
- *The representation for all the input/output objects and the objects defined during the execution of the algorithm is analytically consistent (approximately consistent).* □

The notion of representation and model and the concept of tolerance restriction, and consistency with respect to a theory are used in this section to formally define robustness of geometric algorithms. The addition of a tolerance restriction requirement, which is the basic characteristic of tolerance-based robustness approach, makes the definition of analytical robustness stronger than definition 3. However, as shown later in Section 3, the analytic model definition may be too strong for practical problems with very complicated topology (especially with overconstrained objects) and for problems with higher degree curves and surfaces, in which a less restrictive model definition, the approximated model can be more useful as it will be discussed in Section 4.

Models preserve the properties required in the algorithms. Because the analytic model has the analytic form specified in its representation, it preserves all the properties requested in the theory of the type of objects represented. However, by its definition, an approximated model does not preserve the analytic form of its representation. For example, the approximated model of a line could be a higher degree curve as long as the curve satisfies the tolerance restriction as well as the other constraints. The exact mathematical representation form (e.g., the degree of the curve as a model of a line) of an approximated model is not specified in the definition. However a certain set of properties may still be required by the application. How this is realized is discussed in section .

3 The Analytic Model Method

In this section the first robustness method, the *analytic model method*, is developed. The method consists of the following data structures and mechanisms:

1. A tolerance environment (representation), consisting of three regions for each geometric object.
2. A definition of geometric relatios based on this tolerance environment.
3. A set of tolerance updating rules for updating the tolerances after computing a relation to ensure analytical robustness.

The method will ensure the existence of an analytic model whenever a new relation or constraint is detected or defined to guarantee analytical robustness, according to definition 11. This section will mainly consider linear objects (points, lines and planes). The extension for nonlinear objects (curves and surfaces) will be briefly mentioned at the end of this chapter.

3.1 Dynamic Tolerance Environment

In definition 8, an initial tolerance τ is introduced representing an error bound for the numerical data of the representations, an thus, their uncertainty areas in which valid models may exist. However, as shown in section 2.1, a single and static tolerance region may lead to inconsistencies in the tolerance based decisions made by an algorithm. Here we introduce a dynamic tolerance based representation of geometric objects, consisting of three tolerance regions and then define relations, such as, incidence (coincidence), apartness and ambiguity for this representation.

Definition 12 *(tolerance environment)*

The tolerance environment of a simple object consists of three regions: the ε region, the δ region, and the Δ region, which are r-regions, in the sense of definition 8. The ε, δ and Δ values are initialized as follows: $\varepsilon = \tau - \nu$, $\delta = \tau + \nu$ and $\Delta = +\infty$ (see Figure 7), where τ is the initial tolerance, and ν is a secondary error bound, interpreted as the error in computing relations among geometric objects (we assume that $\tau \gg \nu$). $\square$

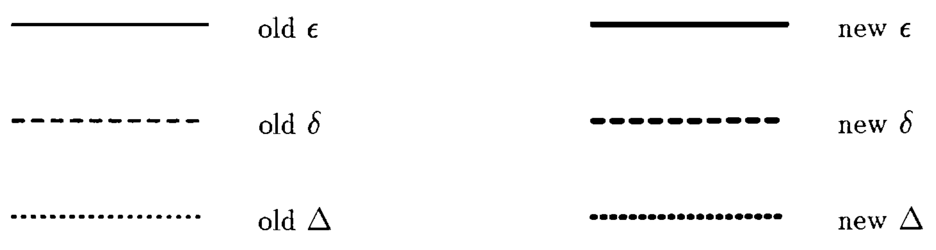


Figure 6: Line types for drawing regions

The ε and δ of the above definition is to be considered as the lower and upper bounds of the initial error bound τ with a secondary error ν). The ε region is therefore interpreted as the region containing all analytic model s of the representation. The δ and Δ regions are used to separate models of "apart" objects. The region δ minus the ε region contains models that are not guaranteed to satisfy the relations computed. This region is therefore called the ambiguity region.

Dynamic tolerance updating, which will be described in subsequent paragraphs, will be used to guarantee robustness. During the tolerance updating, ε regions and Δ regions will shrink, and δ regions will expand. In Figure 6 different line styles are used to depict the three different regions. Bold lines are used to show the regions after tolerance updating. These line styles will also be used in all the figures of the rest for this paper. The initial tolerance definition for a point object and the general configuration of its tolerance regions (after tolerance updating) are shown in Figure 7. Tolerance regions of a line and a plane are shown in Figure 8 and Figure 9. The symbols ε, δ and Δ are sometimes directly used to denote their respective regions if the meaning is unambiguous.

3.2 Relations Between Simple Objects

The topological dimension of a simple object is defined as the smallest dimension of a space in which the type of object may exist. A point has a dimension of zero, a line has a dimension of one and a plane has a dimension of two. In geometric modeling as well as many other applications, it suffices to define the relationships between two simple objects to be one of coincidence, incidence, intersection or apartness. The robust definition of other relations (e.g., inclusion, parallelism) can be derived from these basic relations.

Definition 13 *(apart, coincident, incident, intersecting, ambiguous)*

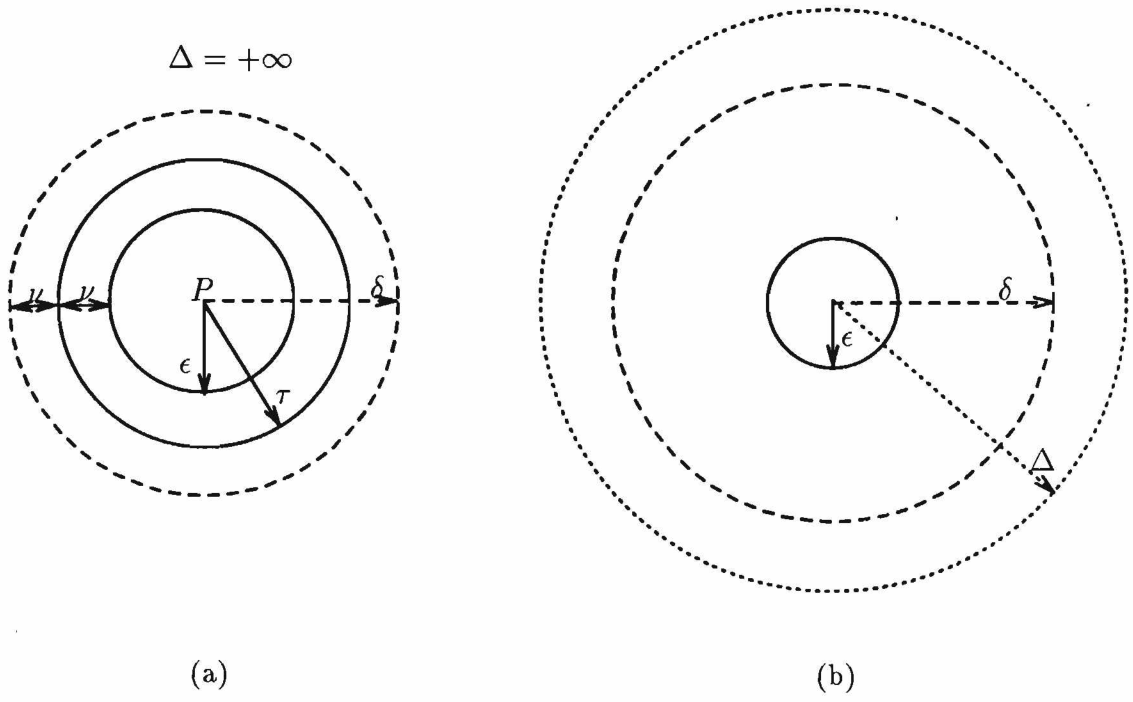


Figure 7: Tolerance definition of a point. (a) Initial tolerance definition; (b) The more general tolerance region positions.

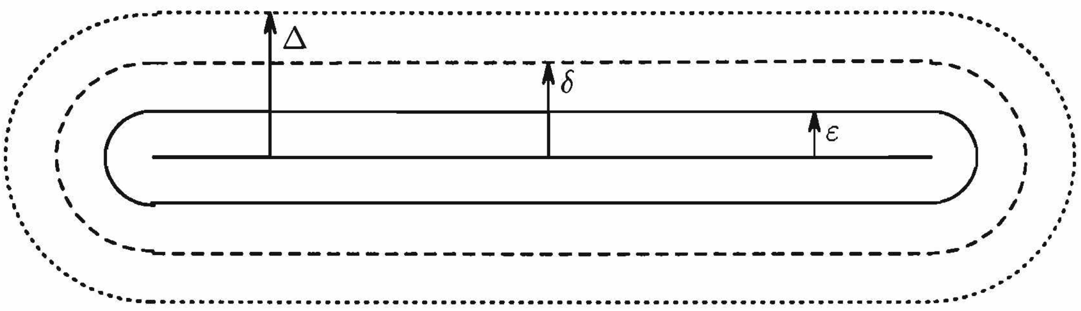


Figure 8: The tolerance regions of a line

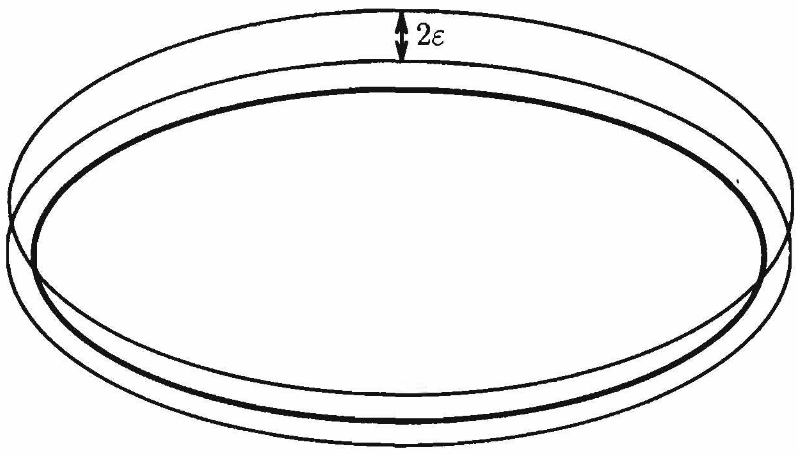


Figure 9: The ε region of a plane

Two simple objects O_1 and O_2 are apart, i.e., $O_1 \neq O_2$, iff their δ regions do not intersect.

Simple objects O_1 and O_2 are coincident, i.e., $O_1 = O_2$, iff they have the same topological dimension and there exists a common analytic model for both O_1 and O_2 within the intersection of their ε regions.

A simple object O_1 is incident on an object O_2, i.e., $O_1 \subset O_2$, iff dimension(O_1) $<$ dimension(O_2) and there exists an analytic model of O_1 in its ε region, say M_1, and an analytic model of O_2 in its ε region, say M_2, so that M_1 is incident on M_2 in Euclidean space.

Two simple objects O_1 and O_2 are intersecting iff their ε regions intersect, and there is no incidence or coincidence relation between the two simple objects.

An ambiguity is detected for a simple object O if the ε region of O is empty or the δ region of O is not a proper subset of the Δ region. $\square$

Figures 10 through 14 show these relations along with their tolerance updating (to be discussed next).

3.3 Tolerance Updating Rules

3.3.1 Two Simple Objects

The following rules are applied to update tolerances after one of the above relations (apart, coincident, incident and intersecting) is detected according to definition 13.

- **rule 1**: If $O_1 \neq O_2$, the radii of their Δ regions will be reduced to half of the minimal distance between the finite parts of the two objects if this distance is smaller than the current Δ value of the simple object, otherwise it is left unchanged (See figure 10.)
- **rule 2**: If $O_1 = O_2$, their representations will be merged into a single simple object O, representing their common analytic models. The new ε and Δ regions of O are defined to be the maximal regions of O inside the intersections of the ε and Δ regions of O_1 and O_2 respectively. The new δ region of O is set to be the minimal region of O enclosing the union of the δ regions of O_1 and O_2 (See figure 11 and Figure 12.)
- **rule 3**: If $O_1 \subset O_2$, according to the definition, they have analytic models M_1 and M_2, respectively, so that M_1 is incident on M_2 in Euclidean space. The ε, δ and Δ regions of the new O_1 and O_2 should be defined as follows (see also figure 13.):
  - the ε, δ and Δ radii of the new representations are the same for both O_1 and O_2.
  - the new ε regions of O_1 and O_2 are the maximal ε regions of O_1 and O_2, respectively, that are included in the intersection of their previous ε regions.
  - the new δ regions of O_1 and O_2 are the minimal δ regions of O_1 and O_2, respectively, that include in the union of their previous δ regions.
  - the new Δ regions of O_1 and O_2 are the maximal Δ regions of O_1 and O_2, respectively, that are included in the intersection of their previous Δ regions.
- **rule 4**: If an intersection relation is computed for two objects O_1 and O_2, and I is the intersection of O_1 and O_2, then their tolerance regions should satisfy the relations $I \subset O_1$ and $I \subset O_2$. The tolerance regions of O_1 and O_2 must then be updated to satisfy the incidence relations according to **rule 3**. Figure 14 shows the tolerance regions for two lines and their intersection point after the tolerance update.

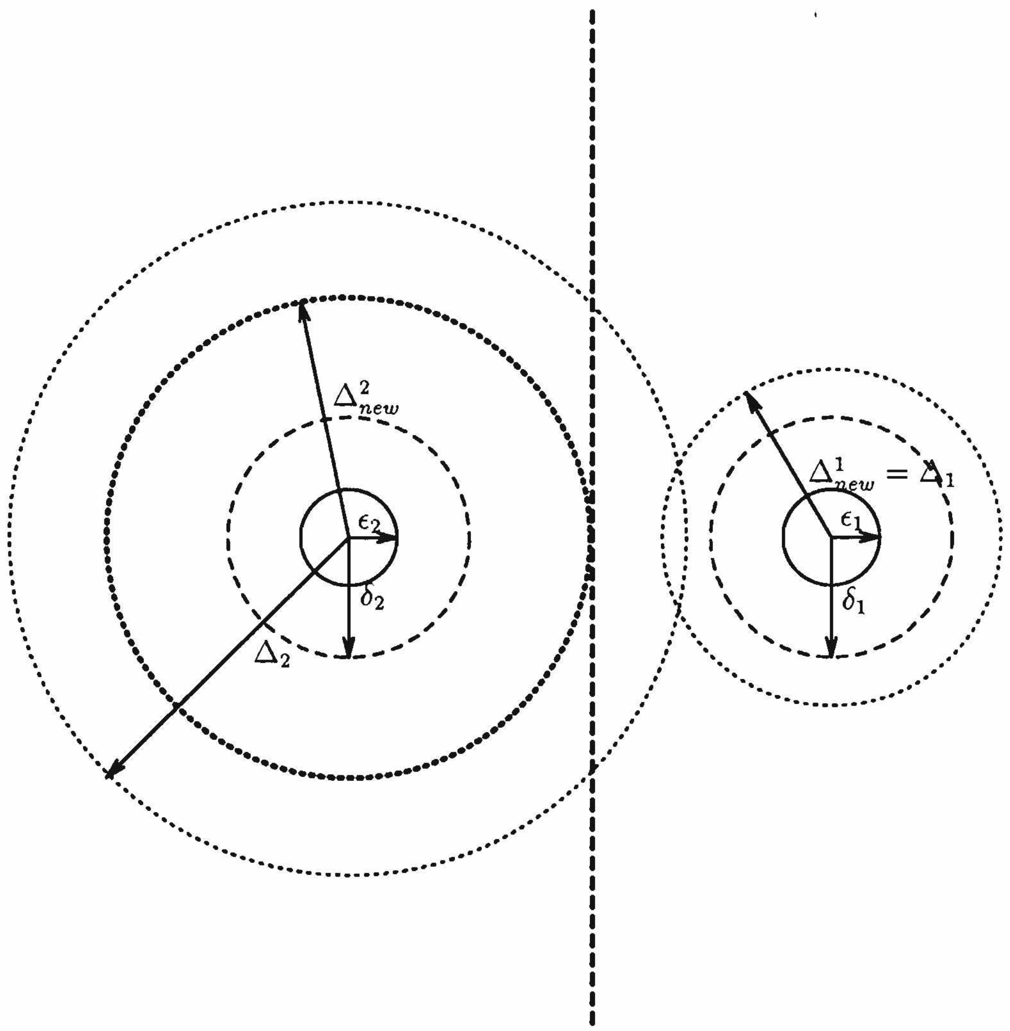


Figure 10: Tolerance update for two apart points

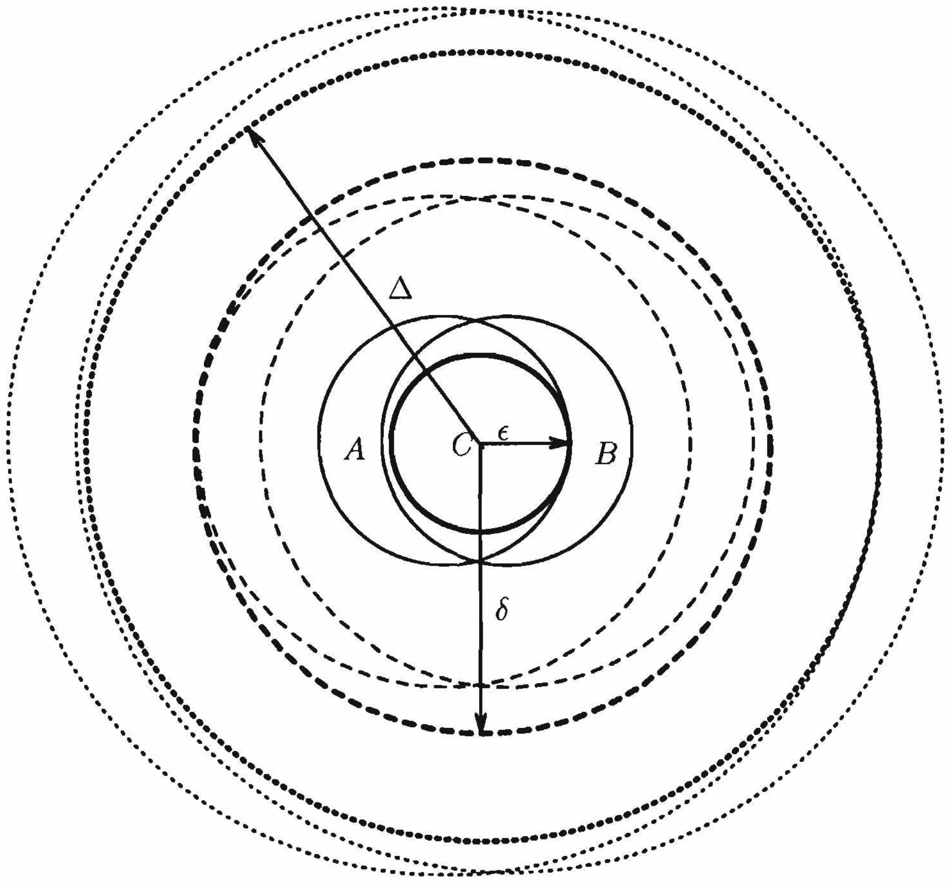


Figure 11: Tolerance update for two coincident points

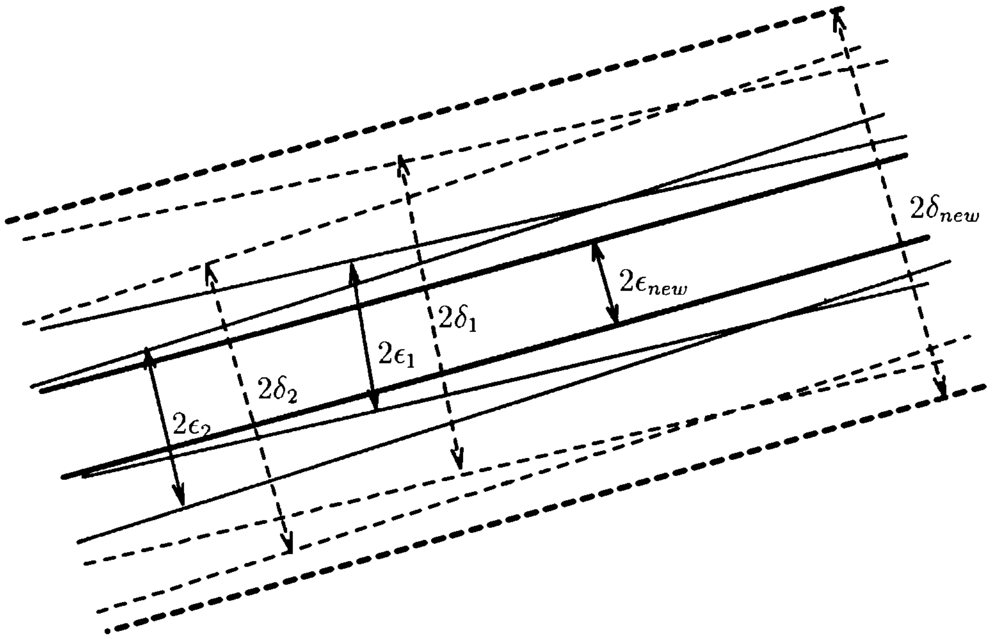


Figure 12: Tolerance update for coincident lines (only ε and δ regions are shown)

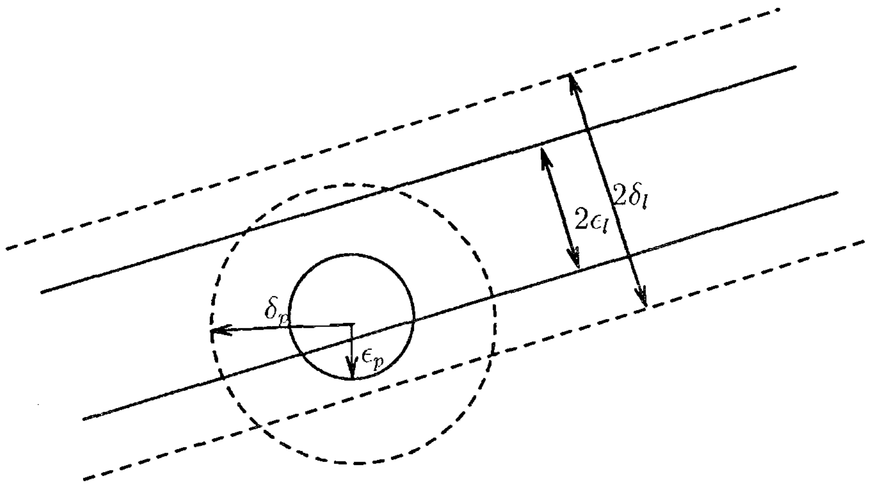
According to the above tolerance updating rules, representations are updated to represent analytic models satisfying the desired relations. The actual implementation of the rules will calculate new numerical data of the representation so that the representation is as close as possible to some real analytic models, that satisfy the relations. In **rule 2**, for instance, two coincident objects are merged and the new representation is calculated to represent a common analytic model of the two objects. The error between the representation and these analytic model s is assumed to be within the secondary error bound ν defined in definition 12. We will call them analytic model the *central models* of the representation.

Definition 14 *(central model)*

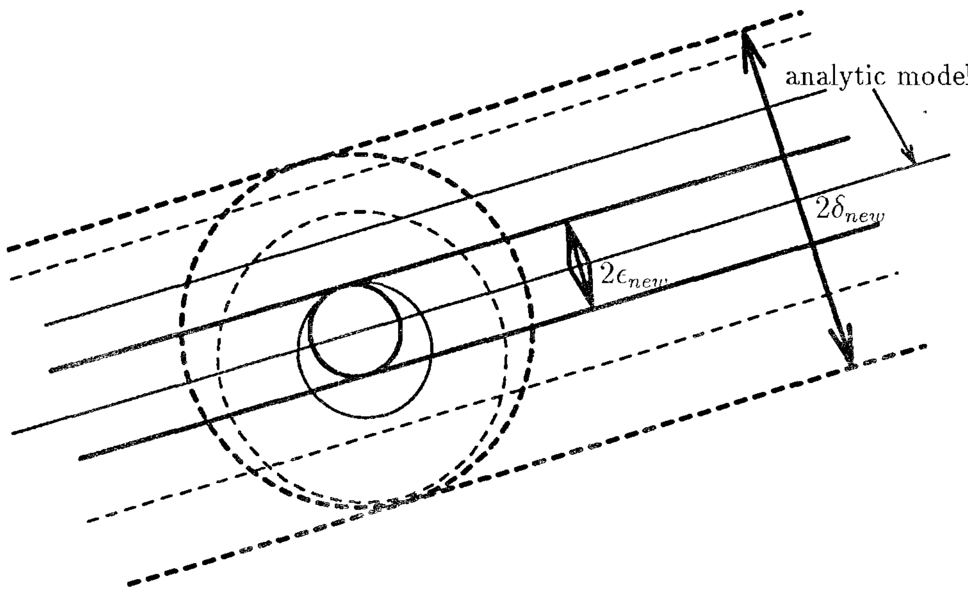
A central model of an object O is an analytic model of O that has a distance of ν or less from the representative of O.

3.3.2 Complex Objects

The four update rules, above, are defined for relations between two isolated simple objects. When dealing with a complex object in which many simple objects are related (directly or indirectly) by incidence and coincidence relations, updating the tolerance of one simple



(a)



(b)

Figure 13: Tolerance update for point-line incidence (a) before update (b) after update.

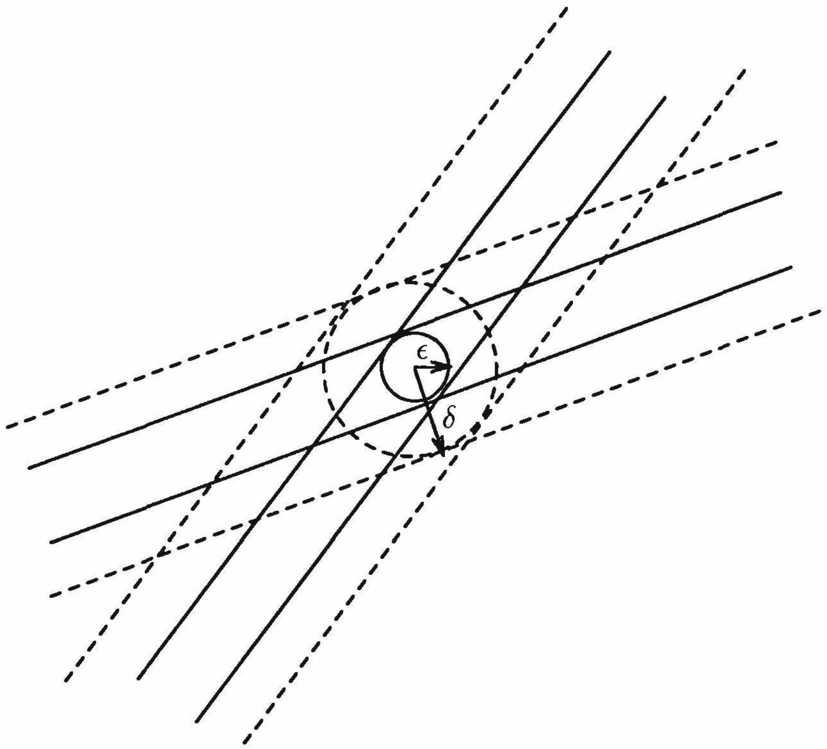


Figure 14: Tolerance update for line intersection (only ε and δ regions are shown)

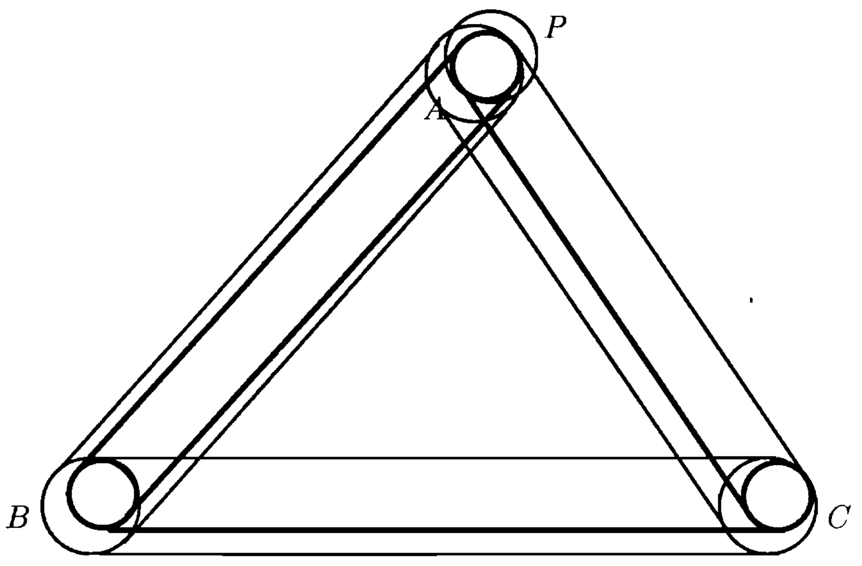


Figure 15: Global perturbation in a complex object (only ε regions are shown)

object generally will affect the tolerances of other simple objects in the same complex object because the change of a simple object might violate the definitions of relations between this simple object and other simple objects in the same complex object. For example in Figure 15 the point P is detected to be coincident to the vertex A of the triangle ABC. Tolerance regions of A and P will be changed. However, in order to preserve the original incidence relations, building the triangle with the three lines and three points, the tolerances of all other lines and points of the triangle must be updated too.

Updating the tolerance of a complex object consists of two parts:

1. Perturbation: the transformation of the tolerance regions by transforming the numerical data of the representation.

2. Changing the tolerance values, i.e., decreasing ε and Δ and increasing δ.

To preserve other relations in the complex object, two rules must be added to the set of tolerance updating rules when comparing two simple objects.

- **rule 5**: When the tolerance regions of a simple object O are updated, all the ε, δ and Δ values of all the other simple objects in the same complex object are set to be that

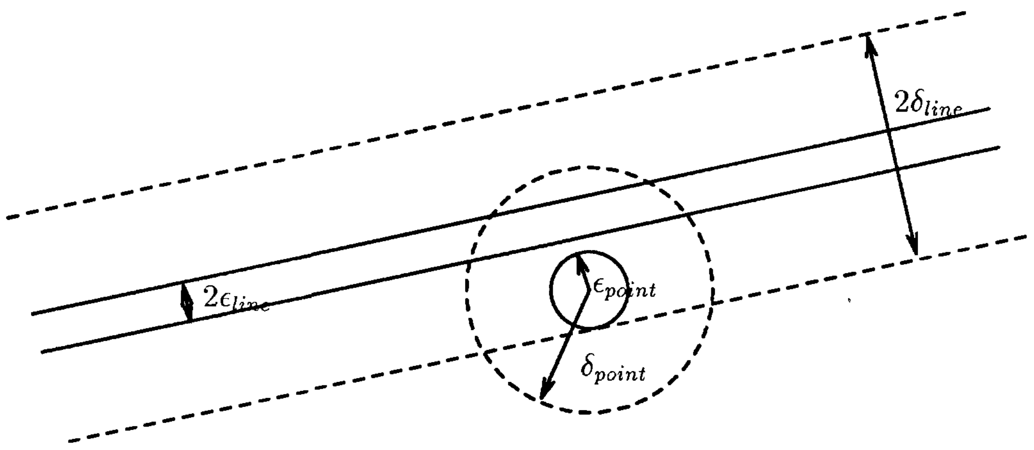


Figure 16: An ambiguous situation

same value. Therefore, all the ε, δ and Δ values of the simple objects in a complex object are always the same.

- **rule 6**: When the tolerance update of a simple object O involves perturbation, the representative of all the simple objects in the complex object must be properly perturbed such that the central models of the representations of these simple objects still satisfy all the relations. Perturbation will be discussed in section 3.5.

When an ambiguity occurs, certain ambiguity handling steps can be taken to solve the ambiguity. Ambiguity handling will be discussed in Section 5. Figure 16 shows an example of an ambiguous situation (the point is neither incident on nor apart from the line).

3.4 Robustness

3.4.1 Properties of Relations

From the view point of intuitionistic geometry, the geometric relations specified in definition 13 should preserve the properties of these relations required in classical Euclidean geometry. The following theorem demonstrates that with a few selected exmaples.

Theorem 2 *(properties of geometric relations)*

If no ambiguities are found in computing the following relations, then

1. *The coincidence relation is an equivalence relation.*

2. *The incidence relation is transitive, i.e., if $A \subset B$ and $B \subset C$, then $A \subset C$.*

3. *If $A = B$ and $A \neq C$, then $B \neq C$.*

4. *If $A \subset B$ and $B \neq C$, then $A \neq C$.*

5. *If $A \subset B$ and $A = C$, then $C \subset B$.*

6. *If $A \subset B$ and $B = C$, then $A \subset C$.*

7. *If two lines A and B intersect or a line A and a plane B are detected intersecting and there are two points C and D, both are incident on A an B (i.e. they are both intersection points of A and B) then $C = D$.*

8. *If planes A and B intersect in two lines C and D, then $C = D$.*

Proof:

1. An equivalence relation must be symmetric, reflexive and transitive. The reflexivity and symmetry follow directly from the definition 13. Now assume $A = B$ and $B = C$, after the tolerance updating, the representations of A, B and C are all merged into one, therefore, the central models of A and C are coincident. From the definition of coincidence in definition 13 follows $A = C$.

2. Using **rule 3**, there exist objects (the central models of A) which are incident on an analytic model of B, which is in turn incident on an analytic model of C. So the central model of A is incident on a model of C. From the definition of incidence in definition 13, $A \subset C$.

3. If we detect $A = B$ first, then it is obvious that $B \neq C$ if $A \neq C$ (since A and B have been merged and have the same tolerance regions). If $A \neq C$ is detected first, then after tolerance updating the δ regions of A and C do not intersect and are inside their Δ regions which have their radii at most half of the distance between A and C. After detecting $A = B$, A and B are merged and have the same ε, δ and Δ regions with the δ region enlarged and Δ region shrunk. Because a δ region is always inside its Δ region, which never grows, and the Δ regions of A and C do not intersect, there

is no possibility that the δ region of the merged object of A and B will intersect the δ region of C, i.e., $B \neq C$.

4. Similar to Property 3.
5. Similar to Property 2.
6. Similar to Property 2.
7. According to **rule 4**, $C \subset A, C \subset B, D \subset A$ and $D \subset B$. After the tolerance updating for these incidence relations, the central models of A and B are all incident on models of C and D. According to Euclidean geometry, each model of C with that property is coincident with the corresponding model of D (two lines, or a line an a plane intersect only once). According to the definition of coincidence in 13 the existence of common models for C and D means that $C = D$.
8. Similar to the proof of property 7. □

3.4.2 Analytical Robustness Theorem

Theorem 3 *A geometric algorithm based on the geometric relations defined in definition 13 and the tolerance updating rules (rules 1 - 6) is analytically robust, if no ambiguity is detected.*

Proof: The way incidence and coincidence relations and the tolerance updating rules for the ε regions are defined ensures that for any two objects O_1 and O_2, related by relation Υ ($O_1 \Upsilon O_2$), for every central model M_1 of O_1 there exists an analytic model M_2 for O_2 (and vice versa) such that the intended relation χ holds for the models: $M_1 \ \chi \ M_2$. For every subsequent relation Υ' computed between an object O_3 and either O_1 or O_2, the same holds true, preserving previous relationships, namely for $O_2 \Upsilon' O_3$ we ensure that for every central model M_2 of O_2, $\exists M_3$ (analytic model of O_3): $M_2 \ \chi' \ M_3$. Therefore $\forall M_1$ (central models for O_1), $\exists M_2 for O_2$ and $M_3 for O_3$ such that $M_1 \ \chi \ M_2$ and $M_2 \ \chi' \ M_3$ (For a case by case analysis, see theorem 2). Now by induction over the number of computed relations, we can conclude that there is an analytic model for the complex object generated by the algorithm (satisfying all the intended relations). According to theorem 3 and definition 11, the algorithm is analytically robust □

3.5 Tolerance Propagation in Complex Objects

When the tolerance of a simple object is updated, the changes will be propagated to the whole complex object as indicated in **rule 5** and **rule 6**. **Rule 5** says that the ε, δ and Δ values of all simple objects in a complex object must be the same. **rule 6** says that all other simple objects must also be perturbed so that their central models satisfy all relations defined for the complex object. It does not, however, specify how to perturb those simple objects to satisfy the relations. This section is dedicated to the approaches to realize the perturbations required in **rule 6**.

3.5.1 Global Perturbation

A simple approach, the global perturbation approach, is to apply the same perturbation to all the simple objects in the complex object, and reset the ε, δ and Δ values to be the same for the tolerance regions of all the simple objects in the complex object as shown in Figure 15.

Attempts to use the global perturbation approach run into difficulties when the two simple objects being compared are in the same complex object. Because the perturbations for the numerical data of the representations of the two simple objects are usually in opposite directions, it is impossible to find a perturbation that will perturb the representations of the simple objects in the complex object so that their central models form a model of the complex object (as shown in Figure 17.) Because in geometric modeling, points, line and planes are usually connected to form the boundaries of solids, global perturbation is only possible in a few exceptional cases or at the beginning of the algorithm process when simple objects have not been connected by any operation.

3.5.2 Local Perturbation

Another simple and sometimes very useful approach is local perturbation. When detecting the relation of two objects that are already in the same complex object, the tolerance propagation can often be localized. The basic idea is to try to perturb those simple objects immediately connected to the simple objects being tested so that the perturbations will not propagate to the whole complex object. Only the ε, δ and Δ values of the rest of the objecst must be changed, and they will be the same for the whole complex object. For instance in

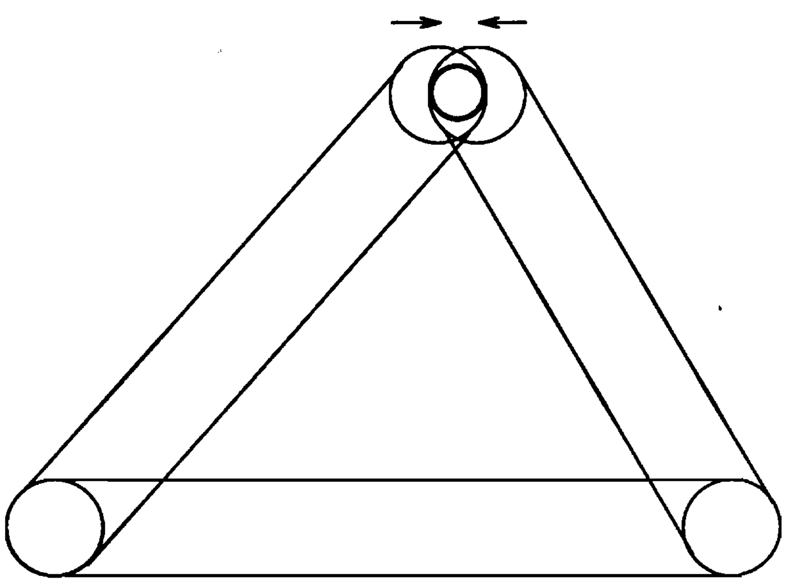


Figure 17: Conflicting global perturbations (only ε regions are shown)

Figure 18, after P_1 and P_2 are found to be coincident and merged to P, tolerances of lines E_1 and E_2 can be updated simply by recreating the lines P_3, P and P_4, P. Thus, the edge E_3 will not be perturbed. In some cases, re-intersection might be necessary as in the case of finding the new position for point P_5 in Figure 18.

The local perturbation approach basically reconstructs part of the complex object to accommodate the perturbation applied to one or more its simple objects. This reconstruction process, however, is sometimes very costly and can become a global operation. For instance, the local perturbation is not possible for the arrangement of the pascal theorem[1] in Figure 3. Any change of one of the nine vertices will have a global effect on all the other lines and vertices (there is a circular dependency between the decisions.)

3.5.3 Zero Perturbation Approach

The approach to be presented in this section, the zero perturbation approach, does not change the numerical data of the representation of any simple object in the complex object at all as if the amount of perturbation needed is zero. Instead, the actual amount of the perturbation

[1]This is an example where the object is over constrained. Some coincidence relations are derived from other relations. However, this can only be shown through complicated theorem proving which is infeasible to be realized as part of computational geometry algorithm.

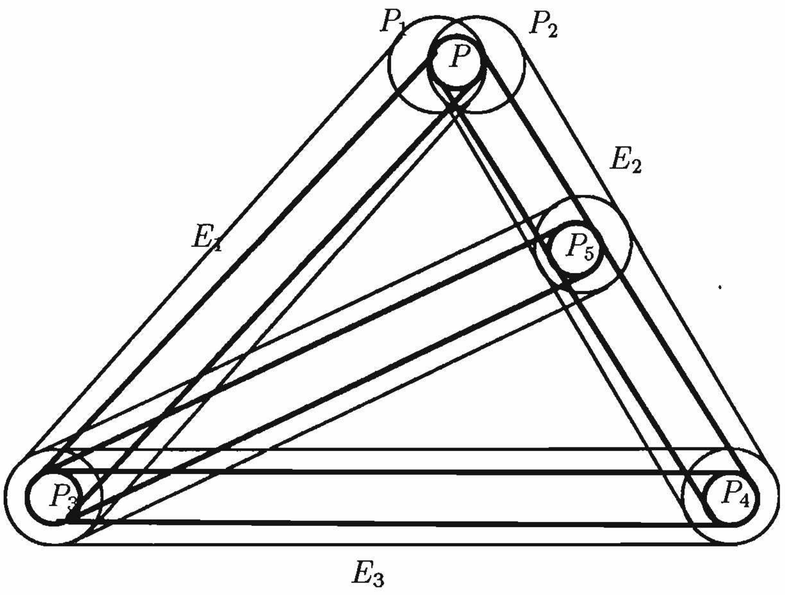


Figure 18: Localize perturbation in a complex object (only ε regions are shown)

is handled as part of the secondary error (computational error of updating tolerances), and the initially defined secondary error bound ν is increased by the amount of the perturbation d. A lemma to be proved next shows that increasing ν by a constant amount for all objects does not affect the consistency of the relations that are already defined.

Lemma 1 *If the r_1 region of point P_1 is contained in the r_2 region of point P_2, then*

- *the $(r_1 - d)$ region of P_1 is contained in the $(r_2 - d)$ region of P_2.*
- *the $(r_1 + d)$ region of P_1 is contained in the $(r_2 + d)$ region of P_2.*

Proof: Regions of points are circles in 2D and spheres in 3D. It is obvious that the lemma is correct for two circles, as shown in Figure 19, as well as for spheres. $\square$

Lemma 2 *If the r_1 region of a point P is contained in the r_2 region of a simple object O and $d < r_1$ then*

- *the $(r_1 - d)$ region of P is contained in the $(r_2 - d)$ region of O.*

- *the $(r_1 + d)$ region of P is contained in the $(r_2 + d)$ region of O.*

Proof: If O is a point, this case reduces to Lemma 1. Therefore, we assume O is either a curve or a surface. Because the r_1 region of P is contained in the r_2 region of O, there must exist a point Q on O so that the r_1 region of P is inside the r_2 region of Q. According to Lemma 1, a simultaneous increase or decrease of the regions of P and Q do not affect the relation of these two regions. □

Lemma 3 *If the r_1 region of simple object O_1 is contained in the r_2 region of simple object O_2 and $d < r_1$, then*

- *the $(r_1 - d)$ region of O_1 is contained in the $(r_2 - d)$ region of O_2.*
- *the $(r_1 + d)$ region of O_1 is contained in the $(r_2 + d)$ region of O_2.*

Proof: We can consider the r_1 region of O_1 as the union of all the r_1 regions of points that are on O_1. For each such point's region, from Lemma 2, increasing or decreasing regions does not affect the inclusion relation of the region of the point and the region of O_2. So the containment relation between the region of O_1, the union of all the regions of points that are on O_1, and the region of O_2 will also not be affected by increasing or decreasing all the regions by d. □

Theorem 4 *If we increase the initially defined secondary error bound ν by d then as long as no new ambiguity is induced, all relations already computed between the objects are not changed.*

Proof: From definition 12, increasing ν by d is equivalent to increasing all initial δ values and decreasing all initial ε values by d. For two coincident or incident objects, as long as no ε region becomes empty after the change of ν, they are still coincident or incident and the new ε and δ regions satisfy the tolerance updating rules for coincidence and incidence relations because of the invariance of region inclusion relation under tolerance adjustment (Lemma 1 – Lemma 3).

Notice that Δ regions are not changed by the change of the ν value. For apartness relations, as long as no new δ region grows out of its Δ region after the change of ν, an apartness relation still holds because there is no possibility for the δ regions of two previously apart objects to intersect without crossing the boundary of their Δ regions, which would result in an ambiguity. □

Now let us summarize and apply the zero perturbation approach. Suppose the relation of two simple objects A, B is computed, and they are already in the same complex object C.

- The relation is apartness: The Δ values for the whole complex object are set to be the smaller of the updated Δ values of A and B (No perturbation of the representatives is needed).
- The relation is coincidence or incidence: Assume that if using the tolerance updating rules in section 3.3, A and B would become A' and B' after perturbations, and ε', δ' and Δ' would be the updated tolerance values, and d_a, d_b are the amounts of perturbations from A to A' and from B to B', respectively. Then the actual tolerance updating will not perturb A and B as well as any simple object in C, and all the ε, δ and Δ values of the simple objects in C will be set to be:

$$\varepsilon_{new} = Min\,(\varepsilon' - d_a,\ \varepsilon' - d_b)$$

$$\delta_{new} = Max\,(\delta' + d_a,\ \delta' + d_b)$$

$$\Delta_{new} = \Delta'$$

Two simple examples for comparing two isolated points and two points in a triangle are shown in Figure 20 and Figure 21.

3.5.4 Problems with the Zero Perturbation Approach

In geometric modeling algorithms geometric objects to be compared are almost always connected. When using zero perturbation approach, therefore, the error in computing relations will accumulate rapidly, which can cause many ambiguities. For example, when comparing two simple objects in the same complex objects (as the two points in Figure 21) over and over again, ε becomes smaller and smaller, and δ becomes bigger and bigger. Eventually, either the ε region will be empty or the δ will exceed Δ causing an ambiguity. In other words, computing relations using the zero perturbation approach is not idempotent. The zero perturbation approach trades a higher probability of detecting an ambiguity for speed and convenience, and the guarantee of obtaining a consistent result. The *approximated model* *method* to be discussed in Section 4 will be even more efficient, however, with a less restrictive model definition, but with less ambiguities.

3.6 Handling Nonlinear Objects

The *analytic model method* is not restricted to linear objects only – it can be generalized to nonlinear objects (curves and surfaces). The following two properties of the curves and surfaces are required for them to be used in the *analytic model method.*

- It can be represented in an analytic form.
- A model exists as long as the ε region is not empty. Here the model is an instance of the object with a specific set of shape parameters.

4 The Approximated Model Method

The *analytic model method* of Section 3 has been shown to have two major problems:

1. The tolerance propagation problem in complex objects. The zero perturbation approach seems to be the only practical approach. However, because the tolerance updating in this approach is not an idempotent operation, often too many ambiguities are created.
2. Complicated curves and surfaces (e.g., general implicit and parametric curves and surfaces) are difficult to implement using this method.

In this section the *approximated model method*, which achieves approximated robustness, will be developed.

4.1 Relations of Two Simple Objects

As in the *analytic model method*, the relation between two simple objects is defined to be either apart, coincident, incident or intersecting. The following definition is similar to definition 13 except that approximated models instead of analytic models are used.

Definition 15 *(apart, coincident, incident, intersecting, ambiguous)*

Two simple objects O_1 and O_2 are apart, i.e., $O_1 \neq O_2$, iff their δ regions do not intersect.

Simple objects O_1 and O_2 are coincident, i.e., $O_1 = O_2$, iff they have the same topological dimension, and there exists a common approximated model for both O_1 and O_2 within their ε regions.

A simple object O_1 is incident on an object O_2, i.e., $O_1 \subset O_2$ iff dimension$(O_1) <$ dimension(O_2), and there exists an approximated model of O_1 in its ε region, say M_1, and an approximated model of O_2 in its ε region, say M_2, so that M_1 is incident on M_2 in Euclidean space.

Two simple objects O_1 and O_2 are intersecting iff their ε regions intersect, and there is no incidence or coincidence relation between the two simple objects.

An ambiguity is detected for a simple object O if the ε region of O is empty or the δ region of O is not a proper subset of the Δ region. □

The above definition of geometric relations is weaker than the definition for their corresponding objects in Euclidean space. For instance, three non-colinear points can be the models of points incident on a straight line, as long as they are within the ε region of that line. On the other hand, because geometric algorithms are all based on Euclidean geometry, in order to be correctly used in geometric algorithms these geometric relations must obey theorems of Euclidean geometry.

4.2 Tolerance Updating Rules

4.2.1 Two Simple Objects

Following is the set of tolerance updating rules for the *approximated model method.*

- **rule 1:** If $O_1 \neq O_2$, the radii of their Δ regions will be reduced to half of the minimal distance between the finite parts of the two objects if this distance is smaller than the current Δ value of the simple object, otherwise they are left unchanged (See Figure 22.)

- **rule 2:** If $O_1 = O_2$, their representations will be merged into a single simple object O, which is a representation of one of their common approximated models. The new ε and Δ regions of O are set to be the maximal regions of O inside the intersections of the ε and Δ regions of O_1 and O_2 respectively. The new δ region of O is set to be the minimal region of O enclosing the union of the δ regions of O_1 and O_2 (See Figure 23.)

- **rule 3**: If $O_1 \subset O_2$, according to the definition, they have approximated models M_1 and M_2, respectively, so that M_1 is incident on M_2 in Euclidean space. O_1 is perturbed by the distance between O_1 and O_2. The ε and Δ regions of O_1 are shrunk so that they are inside the ε and Δ regions of O_2, respectively. O_2 and its tolerance remain unchanged (see Figure 24.)

- **rule 4**: If an intersection relation exists, and I is the intersection of O_1 and O_2, then their tolerances regions should satisfy the relations $I \subset O_1$ and $I \subset O_2$. Tolerance regions of I are defined as: $\varepsilon = min(\varepsilon_1, \varepsilon_2)$; $\delta = max(\delta_1, \delta_2)$; and $\Delta = min(\Delta_1, \Delta_2)$. No tolerance updating is needed for O_1 and O_2.

The rule for the apartness relation is exactly the same as that of the *analytic model method.* Although, from the definition and the rule for coincidence relation, two different kinds of objects (same dimension but different analytic form) can be coincident, in practice, however, two geometric objects with different analytic forms are rarely intentionally considered coincident. A major difference between the rules for the *approximated model method* and the *analytic model method* is the rule for incidence relations after which only the tolerance environment of the lower dimensional object is updated. It has a big impact for the tolerance propagation problem in complex objects (to be discussed next).

4.2.2 For Complex Objects

In the *analytic model method*, tolerance propagation is the major problem affecting its effectiveness. In the *approximated model method*, however, tolerance propagation does not go from the lower dimensional object to the higher dimensional object in incidence relation. Following is the rule for tolerance propagation in complex objects in *approximated model method.*

- **rule 5**: When the tolerance of an object O is updated, all the tolerances of those objects that have been detected to be incident on O also need to be updated so that the incidence relations still hold, i.e., the ε and Δ regions of these objects must be shrunk (if necessary) so that they are inside the ε and Δ regions of O respectively. For example, in Figure 25, the update of the ε region of the curve leads to the updates of the ε regions of all the points on the curves.

Because the change of tolerance of a simple object only affects the tolerances of those lower dimensional objects that are incident on it, tolerance propagation is not a global operation for a complex object. An example is shown in the configuration of Figure 26.

4.3 Robustness

4.3.1 Properties of Relations

Theorem 5 *(properties of relations)*

If no ambiguity is found in computing the relations, then

1. *The coincidence relation is an equivalence relation.*
2. *the incidence relation has transitivity. i.e., if $A \subset B$ and $B \subset C$, then $A \subset C$.*
3. *If $A = B$ and $A \neq C$, then $B \neq C$.*
4. *If $A \subset B$ and $B \neq C$, then $A \neq C$.*
5. *If $A \subset B$ and $A = C$, then $C \subset B$.*
6. *If $A \subset B$ and $B = C$, then $A \subset C$.*

Proof:

The proof of it is similar to the proof of theorem 2.

4.3.2 Approximated Robustness Theorem

Theorem 6 *A geometric algorithm using the relation definitions of section 4.1, the tolerance updating rules of section 4.2 is approximately robust, if no ambiguity is detected.*

Proof: The proof is similar to the proof of theorem 3 except that we use approximated models here instead of analytic model.

4.4 Preserving Other Properties

As shown in section 4.2.2, the *approximated model method* does not have the global tolerance propagation problem the *analytic model method* does. Fewer properties, however, of the geometric relations are preserved in *approximated model method* than in *analytic model method.* Although it is not realistic to require that all properties of the geometric relations be satisfied (Milenkovic[18] pointed out that only an infinite precision algebraic system will always satisfy every theorem of Euclidean geometry) it is still desirable that certain properties should be satisfied in some applications.

For example the last two properties in theorem 2 for the analytic model method are not preserved under the *approximated model method.* In Figure 27, two lines are shown to be intersecting, and two apart points are found incident on both lines. This contradicts the 7th property in theorem 2. This property might be used by some geometric algorithms and therefore should be preserved by additional tolerance updating rules. The following **rule** 6, as an example, is designed to preserve the property that two lines only intersect at one point.

- **rule** 6: When a point P is detected incident on a line L, L will be added to a list (called the parent list of O; the parent list of each point is stored together with this point.) Whenever the parent list of P is updated by adding new lines, the δ value of P will be increased (if necessary) so that P's δ region encloses the intersection area of the ε regions of every pair of lines in the parent list of P as shown in Figure 28. The parent lists of two points are also merged when merging two coincident points.

Theorem 7 *After computing geometric relations and updating the tolerance regions according to rules 1 - 6 of the* approximated model method, *if no ambiguity is found, then the relations in the* approximated model method *also satisfy the following property:*

- *If lines A and B intersect in two points C and D, then C and D are coincident.*

Proof: If C and D are apart, then according to the definition of the apartness relation, their δ regions do not intersect. However, from **rule 6**, both δ regions enclose the intersection of the ε region of A and B. Therefore it is not possible that the δ regions of C and D are separate. i.e., C and D can not be apart. Since the relation is unambiguous, the two points are coincident. □

5 Ambiguity Handling

5.1 The Problem of Ambiguities

In both the *analytic model method* and the *approximated model method*, when one of the following two conditions is satisfied, a so called ambiguity is detected:

- The ε region becomes empty. This happens because sometimes two objects cannot be separated (their δ regions intersect) but their ε regions do not intersect. Deciding it to be any other relation (incidence, coincidence or intersecting) will lead to an empty ε region, which means that possibly no model exists.
- The δ value is greater than the Δ value. This happens because δ regions are always expanding whereas Δ regions are shrinking, at some point the δ region may grow out of Δ. When this happens, the δ region may actually intersect the δ regions of some other objects that have been detected to be apart with this object, which violates the definition of the apartness relation and the properties that are supposed to be preserved by the properly defined geometric relations.

When an ambiguity is detected, it means that we are no longer able to keep the geometric relations consistent with the current tolerances. In order to resolve the ambiguity, tolerances have to be changed, yielding another set of geometric relations. Hopefully the new relations will be unambiguous. If not, we have to continue adjusting the tolerances until a consistent (unambiguous) set of relations is found.

Two questions araise from the above ambiguity solving strategy.

1. Will the tolerance adjustment process terminate? In other words, what happens if we are never able to find a consistent set of relations? In the extreme case, where we define the initial tolerance τ to be large enough so that all the geometric objects are considered coincident, then there will be no ambiguity. i.e., the tolerance adjustment process will certainly terminate eventually when the τ reaches a large enough value, however, the result will probably be useless for all practical purposes.
2. How should tolerances be adjusted? A "lazy" approach is to totally reset the algorithm and redefine all the tolerances. In other words, rerunning the program is necessary every time an ambiguity is found as realized in [2]. This approach is obviously not

optimal because the program may have to rerun many times before it finally reaches a consistent result. It would be much more efficient if we could dynamically and locally adjust the tolerances to solve the ambiguity, and then the algorithm can continue. The rest of this section is devoted to develop such a dynamic tolerance adjustment approach.

5.2 Changing Tolerances

Recall that in section 3.5.2, the zero-perturbation approach uses a dynamic increase of the secondary error ν in a complex object to avoid the global tolerance propagation based on a property of regions described in Lemma 1, Lemma 2 and Lemma 3, i.e., the inclusion relationships among regions will be preserved by a simultaneous increase or decrease of the radii of all regions by the same amount. The same principle can also be used to change the initial error bound τ of all the objects involved while still preserving their relationships.

Theorem 8 *For a complex object, if we increase or decrease the values for ε or δ by the same amount for all simple objects in the complex object, then as long as no new ambiguity is induced, all previously computed geometric relations involving these objects are preserved.*

Proof: The proof follows directly from Lemmas 1, 2, and 3.

Notice that the Δ regions are not changed by these tolerance adjustment. For apartness relations, as long as no new δ region grows out of its Δ region after the tolerance adjustment, an apartness relation should still hold because there is no way for the δ regions of two previously apart objects to intersect without crossing the boundary of their Δ regions. □

5.3 Solving Ambiguities

An ambiguity handling approach can be derived directly from theorem 8.

Ambiguities are caused either by comparing two simple objects or the ambiguity handling itself creates new ambiguities (see below). When an ambiguity occurs for a simple object O, the initial error bound τ of all the simple objects in the same complex object should be adjusted with the following rules, and then the algorithm can continue afterwards.

1. If $\delta \not\subset \Delta$, decrease the initial τ value (equivalently, shrink all the current ε and δ regions) by a value γ, where $\gamma > \delta - \Delta$.

2. If ε is empty, increase the initial τ value (equivalently, expand all the current ε and δ regions) by a value γ, where $\gamma > 0$.

This tolerance adjustment process itself can create new ambiguities. For example, when $\gamma > \varepsilon$ in case 1 above, or when $\gamma > \Delta - \delta$ in case 2 above, new ambiguities are generated. Sometimes, we simply cannot find an appropriate γ value without creating new ambiguities. Also some applications do not allow certain adjustments of their tolerances because of the nature of specific problems (e.g., if high accuracy is required, τ should not be too big). In some of these cases, dynamic tolerance adjustment might not work and a global tolerance resetting and a total rerun of the program may still be possible and necessary.

6 Conclusions

Generally speaking, to achieve robustness, we have to pay the price in other aspects of the geometric algorithms. The algorithms will be somewhat slower because of the computation and updating of tolerances and the ambiguity handling. The accuracy of the algorithm is restricted by the ε regions – no features smaller than the ε can exist.

It is the authors' opinion that ambiguities are an inevitable consequence of tolerance-based approaches. Introducing tolerances means that we acknowledge the fact of inaccuracy and we are willing to sacrifice our capability of interpreting the representation with respect to the intended geometric world to avoid inconsistencies. At the same time, we still want to rely on the properties of Euclidean geometry because the correctness of computational geometry algorithms are based on this assumption. The introduction of ambiguity as a third relation, between incidence and apartness acknowledges the gap between the real world (with computational errors and tolerances) and the ideal world of Euclidean geometry.

A major concern with respect to ambiguities is whether they occur too frequently. Fortunately, from our experiments with the implementation of the two methods [3][7], ambiguities only happen when the relations among geometric objects are indeed ambiguous. For example many objects are very close with a variety of small distances (small features) with a size of about the tolerance τ. However, this kind of ambiguous situation can very often be avoided in practical problems, and a reasonable tolerance definition usually will do the job. Only with problems containing very many small features will this robustness method require corrections, such as tolerance updating, or rerunning the algorithm with new tolerances. This is again inevitable because small features are indistinguishable from errors, and arbitrary decisions have to be made. An alternative possibility would be to apply higher precision floating point arithmetic to resolve ambiguities when they occur, instead of increasing the tolerances.

7 Acknowledgments

The term "Intuitionistic Geometry" was suggested to us by E. Engeler, in a private communication at the Mathematikdepartement at ETH Zurich. 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] ALLEN, G. Testing the accuracy of solid modellers. *Computer Aided Engineering* (June 1985), 50–54.

[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 simlicity: 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., 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.

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

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

[9] 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).

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

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

[12] HOFFMANN, C. M., HOPCROFT, J. E., AND KARASICK, M. S. Towards implementating robust geometric computations. In *Proc. of 4th ACM Symposium on Computational Geometry* (June 1988), pp. 106–117.

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

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

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

[16] 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.

[17] LIANG, Y. D., YE, X., AND FANG, S. Smoothing solids by bicubic Bézier patches. In *Eurographics88, Nice, France* (September 1988).

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

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

[20] MILENKOVIC, V. Calculating approximate curve arrangement using rounded arithmetic. In *ACM Annual Symposium on Computational Geometry* (1989), pp. 197–207.

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

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

[25] 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.

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

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

[28] TROELSTRA, A. S. *Constructivism in Mathematics : An Introduction.* Elsevier Science Pub. Co., 1988.

[29] 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.

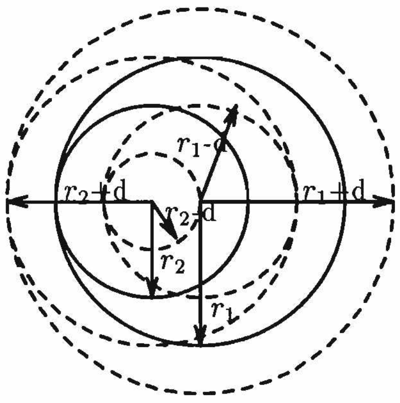


Figure 19: Adjusting the regions of two points while still preserving their containment relation

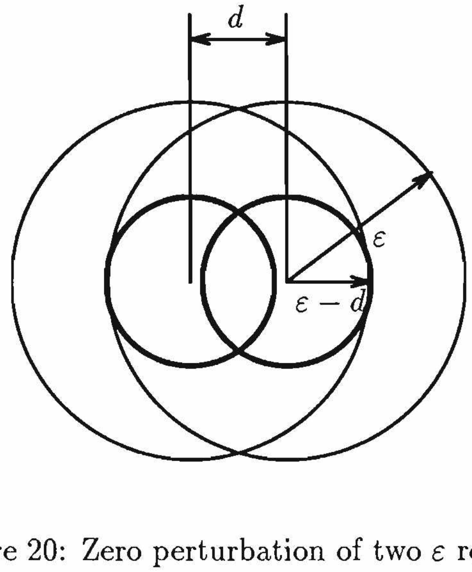


Figure 20: Zero perturbation of two ε regions

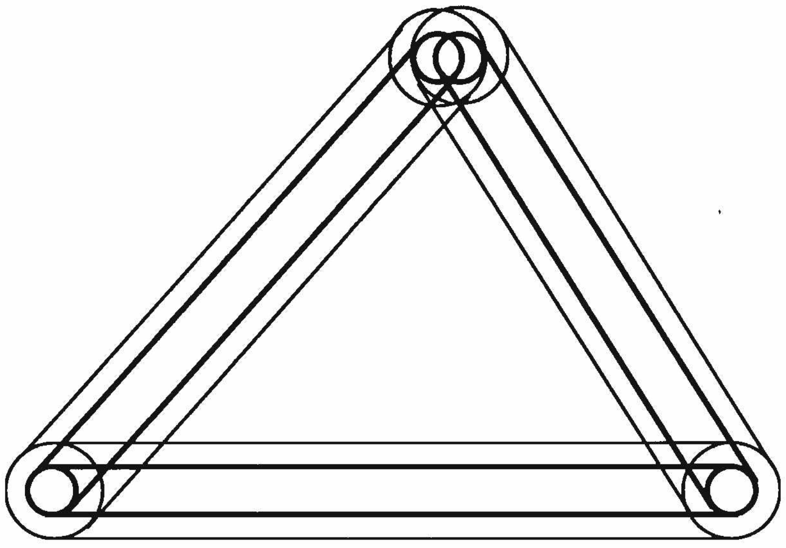


Figure 21: Zero perturbation in a triangle (only ε regions are shown)

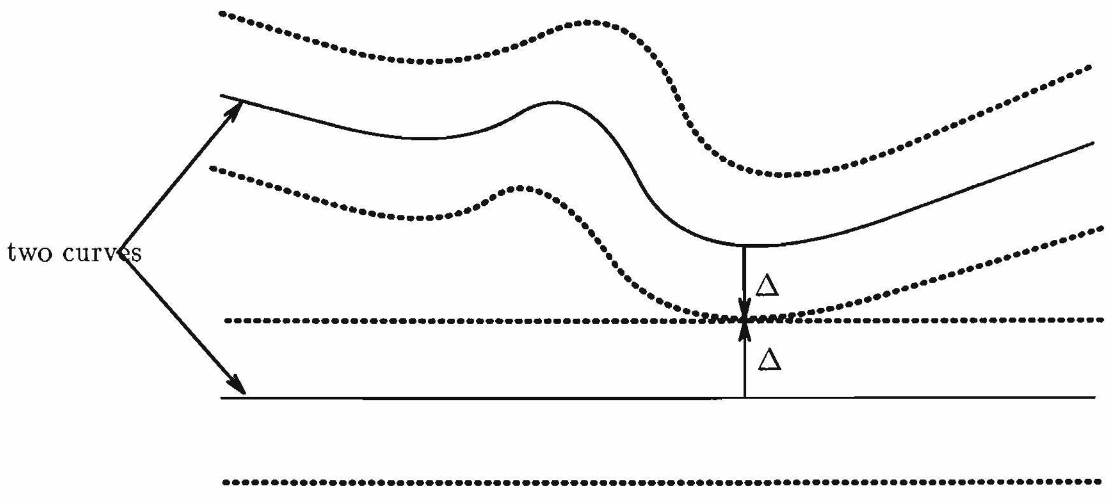


Figure 22: Tolerance update for apart curves (only Δ regions are shown)

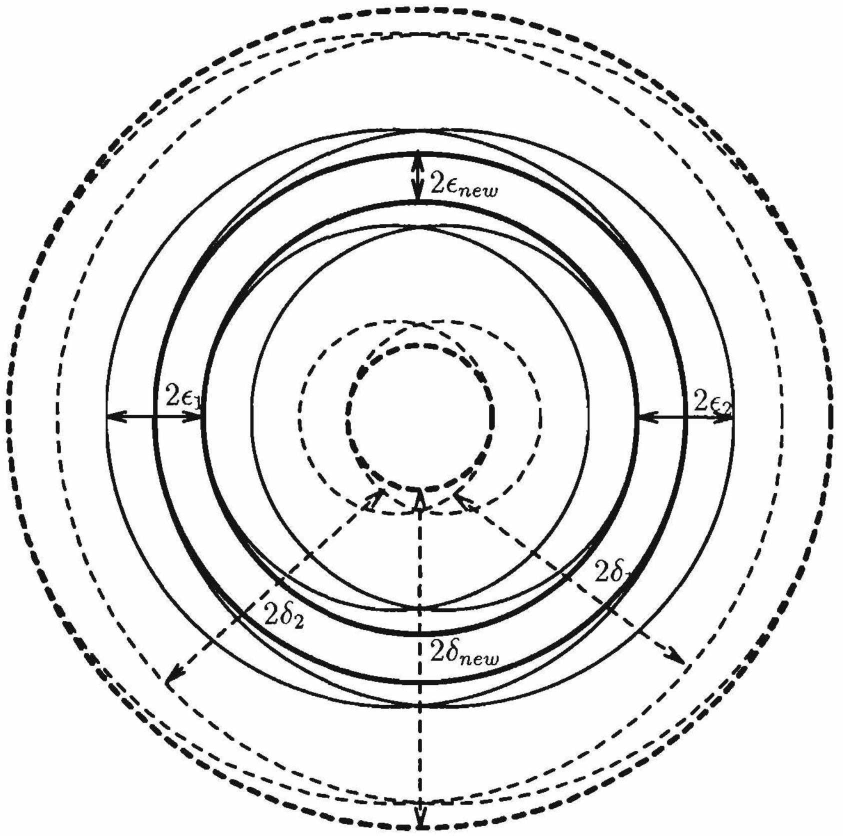


Figure 23: Tolerance update for coincident circles (only ε and δ regions are shown)

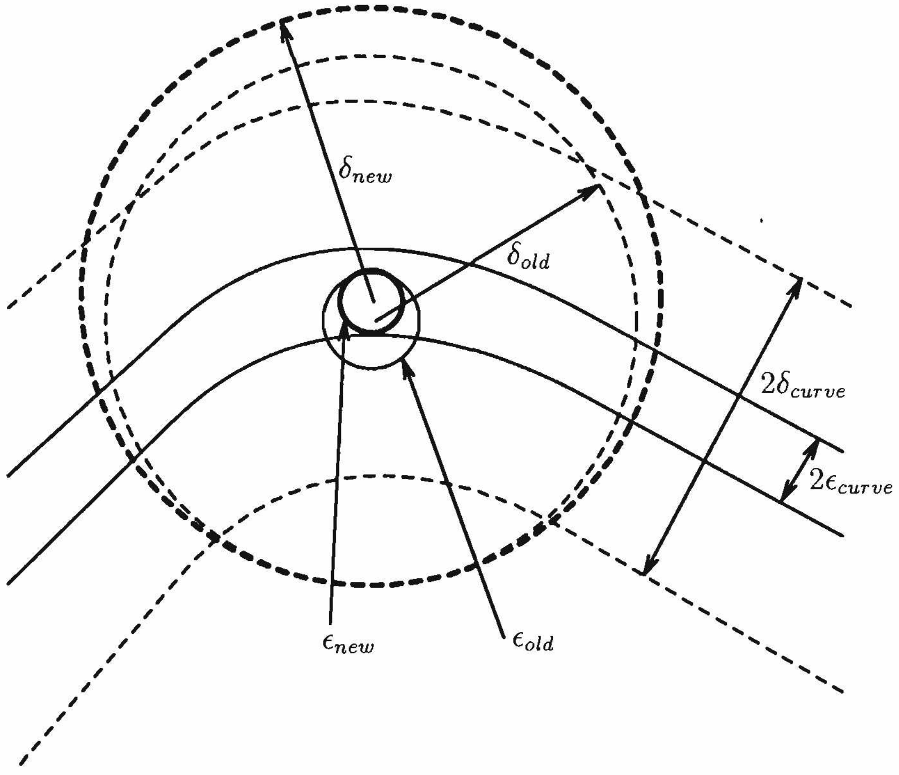


Figure 24: Tolerance update for point-curve incidence (only ε and δ regions are shown)

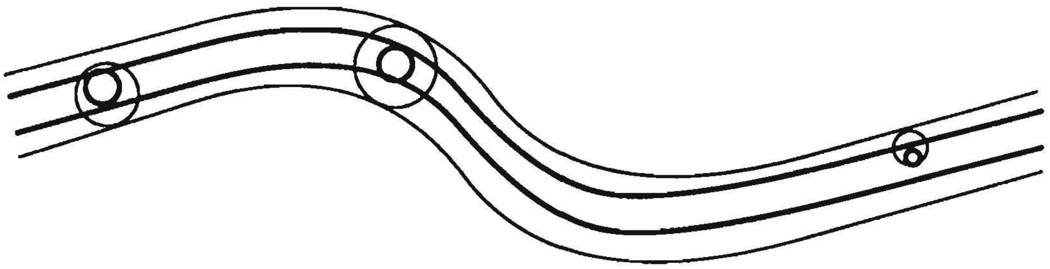


Figure 25: Tolerance propagation in *approximated model method.*

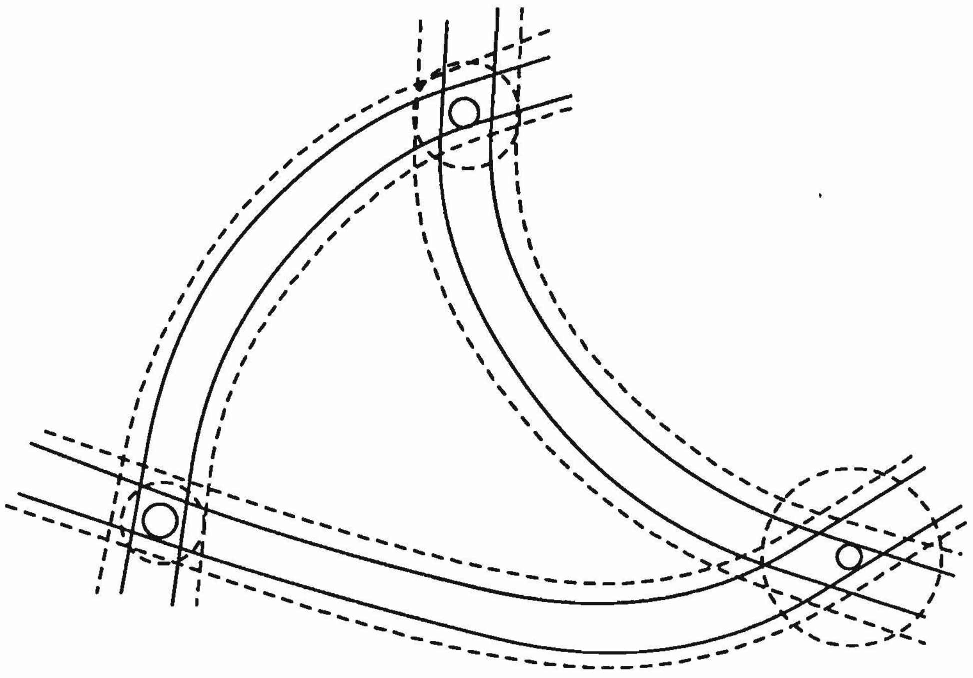


Figure 26: Localized tolerance updating of a complex object

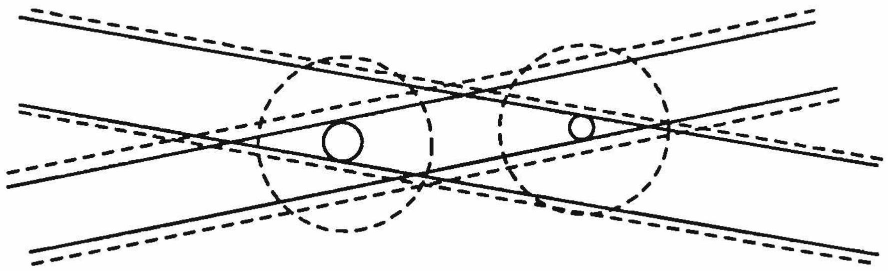


Figure 27: Incorrect intersection of two lines

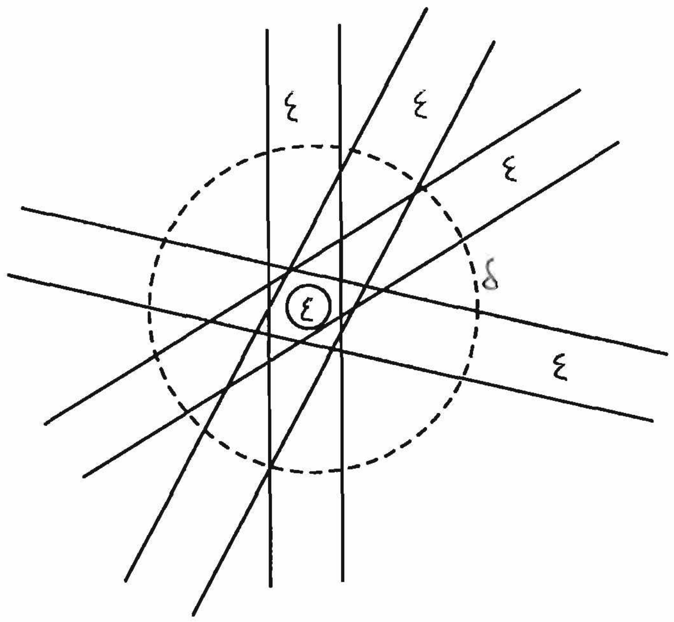


Figure 28: Tolerance of a point with several parent lines