Accuracy and semantics in shape-interrogation applications

C. M. Hoffmann[1] and N. F. Stewart[2]

Revised version

November 22, 2004

Mailing and e-mail addresses

Corresponding author:

N. F. Stewart
Dép't IRO, Université de Montréal
CP6128, Succ. CentreVille
Montréal, Qc, H3C 3J7
Canada
stewart@iro.umontreal.ca

C. M. Hoffmann
Department of Computer Science
Purdue University
West Lafayette, IN 47907-1398
U. S. A.
cmh@cs.purdue.edu

[1]Department of Computer Science, Purdue University; partially supported by NSF grants DMS-0138098 and CCR-9902025.

[2]Département IRO, Université de Montréal, CP6128, Succ. CentreVille, H3C 3J7, Canada. The research of the second author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada, and by NSF grant DMS-0138098.

Abstract.

This paper describes two important questions of semantics, in the context of accuracy in shape interrogation. The first of these questions is how to give meaning to an internally inconsistent solid description based on the widely used trimmed-surface boundary representation. The second question is the meaning of a request, made to a numerical method, to find a solution to a problem whose parameter values are uncertain. Answers to these questions are given, inspired in the case of the second question by a now-standard approach in numerical analysis.

1 Introduction

Shape interrogation is the process of extracting information from a geometric model [1]. In the context considered here, *shape interrogation* may be taken as a synonym for *geometry processing*, a term used earlier by Barnhill [1, *Preface*]. Typical examples of shape interrogation are "given two objects, where is their intersection?", or "given a surface, what does its offset look like?". We are concerned here with the semantics of representations of objects, and with the accuracy of solutions to shape-interrogation problems.

The problem of semantics of representations is disguised, in the questions quoted above, by the use of the word "given". The difficulty [2, 3] is that often in practice the meaning of a given computer representation is far from clear: representations are usually internally inconsistent, due to the use of finite-precision arithmetic, and the use of low-degree curves to represent high-degree surface intersections. Similarly, there are fundamental semantic questions about inaccuracy, which may either be due to the ill condition of problems, or to the instability of methods. These sources of inaccuracy are very different, and consequently, so is the correct approach, in each case, to dealing with it.

Another aspect of accuracy, in the context of shape interrogation, is the definition of how it should be measured. It is a surprising fact that the definition of a metric, to be used to measure error, is missing from a large segment of the literature on robustness of methods in solid modeling.

Appropriate use of the concepts of problem condition, method stability, error metrics, and the relation between error and uncertain data, have been under development for several decades [4, 5], and the overall approach has become standard in modern numerical analysis (see for example [6]). In this paper we will attempt to embed the accuracy and semantics questions of shape interrogation into the standard numerical-analysis approach, and to describe the conclusions to which this leads.

The remainder of the paper is organized as follows. In Section 2, a brief

historical discussion is given, to introduce the two major questions of semantics that will be discussed. Section 3 deals with the semantics of inconsistent representations. Following this, Section 4 discusses two questions related to the exactness of geometric data, Section 5 discusses metrics for the measurement of error, and Section 6 deals with the semantics of ill-defined problems. Section 7 is a short section describing the immediate practical consequences of the paper, and Section 8 is a short conclusion.

2 Historical background

What do we mean when we use an internally inconsistent representation of a solid for the purpose of shape interrogation? And what do we mean when we ask a method to solve a problem whose parameters' values have not been exactly specified? These two questions of semantics will be the main subject addressed in this paper. First, however, some background will be given. This brief historical discussion will be didactic, rather than comprehensive: its goal is to motivate our answer to the two questions above.

The first significant study of the problem of attaching meaning to computer representations, in the context of shape interrogation, was contained in a series of technical reports by Requicha and Voelcker in the 1970's. In particular, in the seminal document [7] Requicha observed that we must distinguish between a representation and the subset of R^3 that it represents, since the same class of sets may be represented in several different ways. One of the main representations discussed was Constructive Solid Geometry (CSG), which describes an object in a procedural way, by means of a binary tree. The technical reports, referred to above, also introduced a careful semantics for CSG, based on the Boolean algebra of regular-closed sets. Regularized Boolean operations, for example $S_0 \cap^* S_1 = cl(int(S_0 \cap S_1))$, were introduced[3], along with closure properties, discussions of mathematical hypotheses corresponding to our intuitive idea

[3]Here, cl denotes closure, and int denotes interior, in the topology of R^3.

of "manufacturable", and so on.

Actual implementation of Boolean operations has proved difficult in certain cases. Exact arithmetic has been advocated by several authors as a reliable tool to resolve these troublesome cases, including those with singularities. Moreover, significant work has been done to overcome the intrinsic loss of efficiency when using exact arithmetic. In the context of polyhedral intersection, for instance, Fortune [8] introduced the distinction between constructors and predicates, and developed the concept of determining, at run time, whether exact arithmetic is actually needed. Keyser *et al.* [9] advocate carefully implemented exact arithmetic to resolve singularities for curved solids. As we explain in detail later, these approaches are applicable only when the input can be understood as exact as given, a situation that does not in general apply for boundary representations. Indeed, the ESOLID modeler [9] relies on Boolean-operation inputs with primitives that are assumed to be exactly represented.

The main paradigm for modern commercial feature-based solid modelers can be viewed as an extension of CSG [10, 11, 12, 13], thereby defining a semantics for feature-based operations. However, for many practical reasons, the kernel of these systems does not use a CSG representation, but uses instead a Boundary Representation. Boundary representations too are supported by rigorous mathematics, going back to Euler and Poincaré [14, 15]. In practice, however, there is a serious difficulty: due to the use of finite-precision arithmetic, and the use of low-degree approximations to high-degree intersection curves, we cannot assume a flawless representation. This is true even in the case of planar-faced polyhedral solids, but the problem is much worse in the case of curved surfaces, such as Bézier or Non-Uniform Rational B-Spline (NURBS) surfaces [16]. Thus, one must deal with "gaps" and "overlaps" of boundary elements near their intended joins, and such defects have serious (and expensive) consequences for downstream applications [2]. More flagrant defects, such as missing surface elements, may also occur. These problems are especially serious in the case of data imported from other systems.

A practical reaction to these difficulties is to attempt to construct "Body Healers", that is, software designed to repair defective geometry on a heuristic basis. And, in practice, Computer-Aided Design (CAD) systems do the best they can to resolve representation inconsistencies by using all sorts of auxiliary information, such as "this surface is a blend surface of that type with these pre-defined contact surfaces". Body Healers are, however, very imperfect, and furthermore, it has been shown that healing cracks optimally in a polyhedral surface has high computational complexity [17]. Also, necessary auxiliary attachment information may not always be available [10]. This is especially true in the modern context of CAD systems federated with third-party software components: such a "best-in-breed" strategy does not permit the use of proprietary attachment information to help geometric queries.

In parallel with the practical need to develop workable heuristic ways to deal with the difficulties mentioned, there is a natural tendency to devote effort to the solution of narrow technical issues that clearly must be resolved. For example, the problem of inexpensive and accurate evaluation of polynomials [18] is a fundamental problem in shape interrogation for which good solutions must be found, and we intend nothing pejorative when we refer to this problem as narrow and technical. On the other hand, it seems worthwhile to take a step back from developing workable heuristics, and solving important but fairly narrow technical problems, in order to ask what our overall objective should be. One advantage of doing this is that we may see more clearly what kind of theorems should be formulated. Another advantage is that in certain cases we may recognize that we are trying to do what cannot be done by numerical analysis, and that some other approach is therefore necessary.

3 The semantics of inconsistent representations

An important example of an inconsistent representation, used in shape interrogation, is the standard Boundary Representation using trimmed-surface patches

[19, 20]. We will denote the representation of a solid S, $S \subseteq R^3$, by Δ. Note, however, that since this representation is internally inconsistent, we have not yet specified what that solid is. Indeed, this is the main topic of this section. The notation used here follows [3].

The data Δ is in two parts, the *geometric data* and the *topological data*. The geometric data in Δ comprises a finite set of compact oriented 2-manifolds-with-boundary, and corresponding sets of explicit boundary curves and corner vertices. The underlying surface for each 2-manifold is represented by a spline function over a parametric domain D_0, with range in R^3. The spline function is then restricted to a subset D of D_0, as delineated by certain curves in the parametric domain, yielding a *trimmed NURBS patch*. (Such trimming may arise, for example, as the result of the intersection of two surfaces for representing the boundary of the solid.) Thus, each 2-manifold is represented by a trimmed NURBS patch.

Figure 1 illustrates two trimmed patches which join (approximately) along the intersection of two surfaces $\boldsymbol{F}$ and $\boldsymbol{F}'$, restricted respectively to D and D'. The part of the representation of the solid corresponding to the intended intersection comprises two (almost certainly inconsistent) pre-images, defined by parametric curves $\boldsymbol{p}$ and $\boldsymbol{p}'$ with ranges in the respective parametric domains. (We will follow [19] and refer to $\boldsymbol{p}$ and $\boldsymbol{p}'$ as *p-curves*. Many practical numerical algorithms compute one or both of these two pre-images $\boldsymbol{p}$ and $\boldsymbol{p}'$ as the output for an intersection of surfaces.) In addition, there is often a third representation, usually inconsistent with the other two, which is a parametric curve $\boldsymbol{b}$, with range in R^3, and which follows closely the images of the p-curves $\boldsymbol{p}$ and $\boldsymbol{p}'$. Finally, there is also an explicit (and possibly inconsistent) representation of each endpoint $\boldsymbol{v} \in R^3$ of the parametric curve $\boldsymbol{b} = \boldsymbol{b}(t)$, as illustrated in Figure 1.

The representation Δ also contains symbolic information, or *topological data*, describing how the faces, edges and vertices of the cellular decomposition of the boundary ∂S of S fit together. This data defines a topological 2-cycle: two typical adjacent faces are illustrated in Figure 2. Ideally, the geometric and topological data are consistent: for example, corresponding to each 2-cell [21,

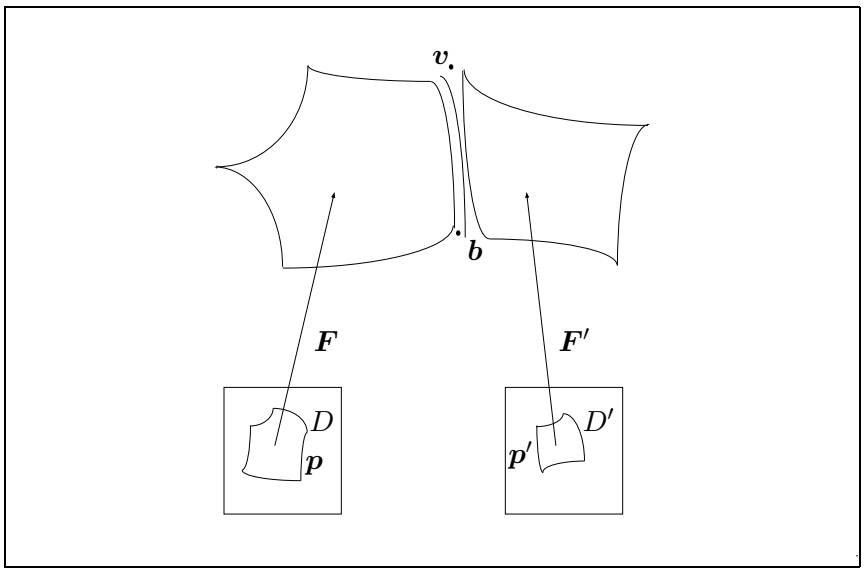

Figure 1: Two adjoining trimmed NURBS patches

Ch. 14] in the topological data is a trimmed surface patch in R^3 (two of these, $\boldsymbol{F}[D] = \{\boldsymbol{F}(u,v) : (u,v) \in D \subseteq D_0\}$ and $\boldsymbol{F}'[D'] = \{\boldsymbol{F}'(u,v) : (u,v) \in D' \subseteq D_0\}$, are shown in Figure 1). As already mentioned, the actual boundaries of these trimmed patches are stored implicitly, by appropriate pre-images (p-curves such as $\boldsymbol{p}$ and $\boldsymbol{p}'$). Ideally, there are two p-curves corresponding to each 1-cell in the topological data, one associated with each adjacent face, as well as one explicit boundary representation $\boldsymbol{b}(t)$ for each 1-cell in the topological data. Also, there should be one explicit corner vertex $\boldsymbol{v}$ corresponding to each 0-cell in the topological data.

Unfortunately, as illustrated in Figure 1, the curve $\boldsymbol{b}(t)$ does not usually coincide exactly with the corresponding edge of $\boldsymbol{F}[D]$, nor with the corresponding edge of $\boldsymbol{F}'[D']$. Similarly, the vertex $\boldsymbol{v}$ may not coincide exactly with the ends of the neighboring boundary curves (such as $\boldsymbol{b}(t)$), nor with the exact corners of the trimmed patches $\boldsymbol{F}[D]$ and $\boldsymbol{F}'[D']$. Finally, in practice, the geometric data may become inconsistent with the topological data.

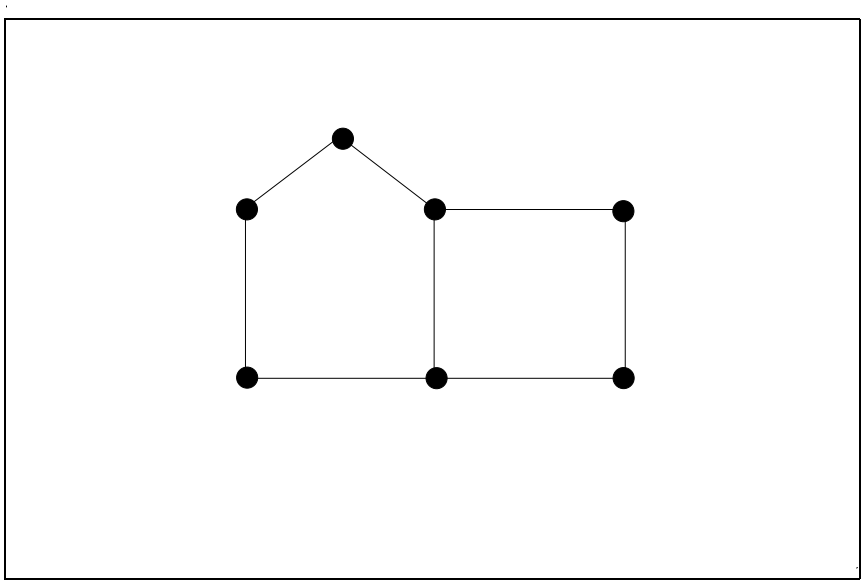

Figure 2: Topological data corresponding to the trimmed patches of Figure 1

If we are going to prove rigorous theorems about the subset of R^3 obtained by applying an algorithm to the representation Δ, then clearly we must assign a meaning to the inconsistent representation Δ. A solution to this problem was presented in [3], where a set S, $S \subseteq R^3$, was assigned to each well-formed representation Δ. This assignment was denoted $S \models \Delta$, read "S is the realization of Δ". We note immediately that there is no requirement to actually compute the set S: we only need to know that it is well-defined, and to have good bounds on its shape, so that we can prove theorems about the results of applying methods to the representation Δ. We must also present cogent arguments to support the claim that S is a reasonable interpretation of the inconsistent data Δ.

The realization S is defined by its boundary ∂S, which is made up of slightly perturbed trimmed NURBS patches from Δ, where the perturbation is defined by the Whitney Extension Theorem. The slightly perturbed patches are not necessarily NURBS patches [3], but they are mutually consistent with the explicit vertices $\boldsymbol{v}$ and with slightly perturbed versions of the explicit boundary curves

$\boldsymbol{b}(t)$, and they all fit together in a way that is exactly consistent with the topological data. Furthermore, the meaning of the words "slightly perturbed" here is quite satisfactory. The perturbation of the trimmed patch is denoted $\boldsymbol{\epsilon}(\boldsymbol{q})$, $\boldsymbol{q} \in D$. It is continuous, and the magnitude of the perturbation nowhere exceeds the magnitude of the largest discrepancy between the edge of the given patch and the given neighboring explicit boundary curves $\boldsymbol{b}(t)$. (In other words, the perturbation is nowhere larger than the largest discrepancy already in the given data, along the edges of the given patch.) In addition, the perturbation satisfies a Lipschitz condition throughout the patch, with a Lipschitz constant for each component γ of $\boldsymbol{\epsilon}$ equal to

$$L \;=\; \sup_{\boldsymbol{q}_1, \boldsymbol{q}_2 \in \partial D,\; \boldsymbol{q}_1 \neq \boldsymbol{q}_2} \frac{|\gamma(\boldsymbol{q}_1) - \gamma(\boldsymbol{q}_2)|}{\|\boldsymbol{q}_1 - \boldsymbol{q}_2\|} \quad .$$

(Thus, the perturbation satisfies a Lipschitz condition, with a constant equal to the constant in the Lipschitz condition corresponding to the discrepancy already present in the given data, along the edges of the given patch.) The Lipschitz condition is important, because it allows us to bound the change in the normal vector of the perturbed patch, relative to the normal vector $\boldsymbol{n}$ of the given trimmed patch. This bound will be necessary in proofs about computed sets, when we want to preclude the possibility that adjacent patches have extraneous intersections (*i.e.*, intersections other than those along their prescribed common edge, or at their prescribed common corner) [22, 23].

The definition of well-formedness of Δ was also given in [3], along with a discussion of verification of whether a given Δ is well-formed and the cost of such verification. (The cost may be negligible, if good sufficient conditions are known, or higher if algorithms like [24] are required [3, Sec. 3.3].) The sets S defined by the slightly perturbed NURBS patches were called QuasiNURBS sets.

Other approaches to assigning a meaning to the inconsistent representation are possible, but they will not be discussed in detail here. One possibility is to introduce the concepts of inner-interval and outer-interval solids [25], obtained by enclosing a solid's boundary in boxes computed using interval arithmetic. For

example, the larger set obtained in this way contains the original set, in the same manner as a Maximum Material Condition (MMC) of geometric tolerancing [26]. In the case where the original solid is a manifold with a well-defined boundary, it is possible to show that the topological structure of the inner-interval and outer-interval solids is the same as that of the original solid. In the case of data corresponding to an ill-defined boundary, as in the case of Δ, above, an outer-interval solid can still be computed using interval arithmetic, and this can be viewed as a representative of Δ. There is still a gap in the semantics, however, since the representative depends on the choice of interval-arithmetic operations.

A different idea for assigning meanings to representations appears in [27], which suggests identification of inconsistent data like Δ with a *class* of subsets of R^3. This is accomplished by defining, by means of a procedural definition, the concept of an ϵ-solid, which is specified in terms of inner and outer sets satisfying additional conditions of regularity. This is a new and original approach.

One of the main advantages of the QuasiNURBS set S, aside from its geometrical similarity to the given geometric data in Δ, is that it has the same topological structure as that specified by Δ. Suppose, for example, that we are given well-formed representations Δ_0 and Δ_1. Then the topological structure of S_0 and S_1, where $S_0 \models \Delta_0$ and $S_1 \models \Delta_1$, as well as the topological structure of sets defined by subsequent operations, such as $S_0 \cap^* S_1$, are all well defined and capable of being referenced in the statement of a theorem. The QuasiNURBS sets appear therefore to provide a satisfactory foundation on which to build a theory of robustness in shape interrogation.

4 Exactness of data

There are two questions related to the exactness of data that have received attention in the field of robustness. One of them has already been dealt with, in Section 3. We discuss it further here because we wish to distinguish it from the second question, which will be considered later in this section.

The first question related to exactness of data is, given demonstrably inconsistent data Δ, should we assume that the topological data is correct, and define the relation $\models$ by slightly modifying the geometric data, or should we assume that the geometric data is correct, and try to make appropriate modifications of the topological data [28, 29]? In [3] and in Section 3, above, the first choice was made. The reasons for this choice were as follows [3]. First, the decision seems to be thrust upon us. Even for solids of very simple form, such as a simple cube with sides that are not quite planar, the edges of $\boldsymbol{F}[D]$ and $\boldsymbol{F}'[D']$ almost certainly do not correspond. In this case, the geometric data is, patently, inexact. Secondly, suppose the user provides, say, a cylindrical surface as input. It may be important in practice that this surface be very nearly cylindrical, but the user does not believe that the eventually manufactured object will be an *exact* mathematical cylinder. Such a user should be prepared to accept that the inconsistent data, which he himself provided, models an object with surfaces varying slightly from cylindrical. Thirdly, in the case when there is uncertainty in the data provided, this uncertain data may correspond to uncertainty in the actual form of the object surfaces.

In the case of imported data, it may be that neither the geometric nor the topological data is reliable. In this case, we again choose to modify the geometric data.

The possibility of uncertainty in the data, however, leads us to the second question concerning exactness of the data. Let us denote the whole collection of geometric data in Δ by $\boldsymbol{g}$: it includes the vertices $\boldsymbol{v}$, the control points for the spline curves $\boldsymbol{b}$ and the spline surfaces $\boldsymbol{F}$, as well as the control points for the p-curves in the parametric domains. The representation Δ is thus parametrized by $\boldsymbol{g}$, and we write $\Delta_{\boldsymbol{g}}$ to refer to the representation defined by the geometric data $\boldsymbol{g}$. The question then is, if the user provides a single value $\boldsymbol{g}_o$ of this geometric data, can it be considered exact, or is there in fact uncertainty in the data, so that the actual problem[4] to be solved is known only to be a member of the *class*

[4]For simplicity, we are considering here the case when the problem is defined by a single set,

of problems

$$\{\Delta_{\boldsymbol{g}} : \boldsymbol{g} \text{ in a neighborhood of } \boldsymbol{g}_o \} \qquad ?$$

The answer to this question has important consequences for the kind of methods that can appropriately be applied. Consider, for example, the former case: if $\boldsymbol{g}_o$ can be considered exact, then, even though the definition of $\models$ involves slight perturbation of the geometric data (Section 3), it makes sense, provided there is no budget constraint, to apply exact arithmetic to solve numerical problems. To take a very simple illustration, let $S \models \Delta$ be defined as in [3] and in Section 3, above. Then, we can compute the translation

$$S + \boldsymbol{t} = \{\boldsymbol{x} : \boldsymbol{x} = \boldsymbol{s} + \boldsymbol{t}, \boldsymbol{s} \in S\}$$

by using exact arithmetic to add $\boldsymbol{t}$ to each vertex of $\boldsymbol{v}$ of Δ, and to each vector control point of each curve $\boldsymbol{b}(t)$ of Δ, and to each control point of each $\boldsymbol{F} = \boldsymbol{F}(u, v)$ of Δ. It can then be *proven*, since spline curves and surfaces are invariant under translation [16], that the resulting *computed* representation Δ_c satisfies

$$S + \boldsymbol{t} \quad \models \quad \Delta_c.$$

(We emphasize that $S + \boldsymbol{t}$ denotes the exact mathematical translation of S.)

In the other case, when there is uncertainty in the given data ($\boldsymbol{g}$ is known only to lie in some neighborhood of $\boldsymbol{g}_o$), the use of exact arithmetic is not appropriate, as will be discussed further in Section 6. In practice, there are many possible sources of uncertainty in the data $\boldsymbol{g}$. For example, there may have been error in the decimal-to-binary conversion, and it is quite possible [3], in ordinary circumstances, that the desired width of an object should be, say, $\sqrt{2}$, a number which has no finite binary representation. Similarly, there may have been error introduced by some previous solid-modeling operation, or the object to be interrogated may have come from some other program or device.

Consider for example the objects shown in Figure 3: S_0 is a block, and S_1 is a wedge-shaped object. The object S_1 is illustrated by itself at the right of the

as for example in set translation.

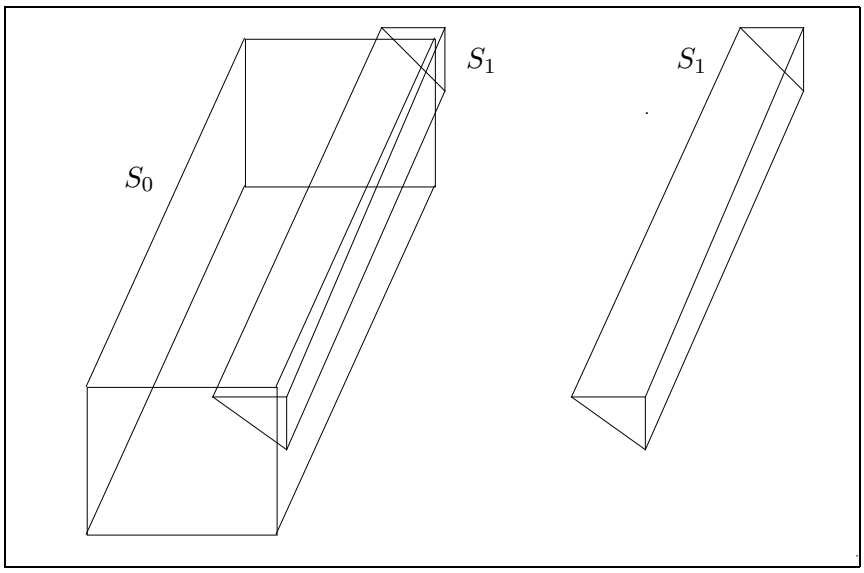

Figure 3: Wedge to be subtracted from a block

figure, but it is supposed to be positioned approximately in the upper righthand part of S_0, as illustrated at the left of the figure. The operation of interest is the set difference $S_0 -^* S_1 \equiv S_0 \cap^* (\sim S_1)$. Certain of the faces of S_0 are almost (but not quite) coplanar with certain faces of S_1, and the method may have no way to know that there is uncertainty in each of the faces of a pair, and that (as a result of external information) the faces should coincide. Such examples can arise, in particular, in the context of data exchange in feature-based systems [10, p. 186, col. 2]. The correct response, in such a situation, is not the application of expensive exact arithmetic in a futile effort to produce a satisfactory answer. (Use of exact arithmetic in this case will produce very thin sets that are entirely spurious.) Rather, we should recognize that the problem is very ill-conditioned, which means that small changes in the problem data can produce a very large difference in the solution (see Section 6). The consequence is that accurate solutions are beyond the reach of improved numerical methods, since we can never avoid the error associated with the uncertainty in the problem

data. The solution to the dilemma is to reformulate the high-level problem so that we avoid the operation of computing $\cap^*$, replacing it by an operation that includes relevant attachment information as part of its inputs [10, p. 186, col. 2].

These questions will be discussed in more detail in later sections, but we will make two remarks here about the example of the last paragraph. First of all, the authors of [10] did not need an elaborate theory in order to conclude that the operation $\cap^*$ should be replaced, as described above (for them it was obvious). But that is not the point. The point is that when we are considering, in the abstract, the question of computing $\cap^*$, the appropriate reaction in the case of a badly conditioned problem is to admit our inability to produce an accurate solution, and conclude that the high-level problem should be reformulated. An inappropriate reaction is to apply expensive methods to solve exactly a problem that is almost certainly not the one we want to solve, and for which small changes in the problem definition may lead to large changes in the computed result.

The second remark is that such a decision, to abandon a problem formulation because it leads to an ill-conditioned problem, is very much within the tradition of numerical analysis. For example, least-squares approximation using the ordinary power basis leads [30, p. 194] to a linear system with coefficients defined by the Hilbert matrix, which is very badly conditioned. The solution to this difficulty is to reformulate the high-level approximation problem using orthogonal polynomials, so that the ill-conditioned matrix problem is avoided.

5 Metrics for the measurement of error

As mentioned in the introduction, it is a surprising fact that the definition of a criterion, for the measurement of error, is missing from a large segment of the robustness literature. For example, a test for robustness of set-intersection algorithms that has been widely used is the following. One input object is taken to be a unit cube, and the other input object is taken to be a slightly rotated copy of the same cube. The robustness of the algorithm is then evaluated, based

on the smallness of the angle of rotation that can be tolerated without a program "crash". The problem with this approach, of course, is that it makes no reference to the quality of the computed result. For example, an algorithm that simply produces the unit cube as its output, independently of its inputs, would be considered ideal according to this criterion [31, p. 4].

On the other hand, it is not straightforward to introduce a metric, on classes of sets, that corresponds to our intuitive idea of closeness in the context of shapes. A first idea is to introduce the Hausdorff metric on the class $\mathcal{C}$ of non-empty compact subsets of R^3 ($\mathcal{C}$ includes the r-sets):

$$d(S, S') = \max\{\sup_{\boldsymbol{s} \in S} dist(\{\boldsymbol{s}\}, S'),\ \sup_{\boldsymbol{s}' \in S'} dist(S, \{\boldsymbol{s}'\})\}.$$

This can be extended to the class $\mathcal{C} \cup \{\emptyset\}$, respecting the triangle inequality, by using the extended real line, defining $d(\emptyset, \emptyset) = 0$ and $d(\emptyset, A) = \infty$ for $A \neq \emptyset$. The Hausdorff metric is a very rough measure: for example, the unit ball is considered close to a unit ball with a thin hole drilled through its middle [32]. A more sensitive measure is the metric

$$d_w(S, S') = \max\{d(S, S'), d(\partial S, \partial S')\}$$

which requires that if two objects are to be considered close, then both the Hausdorff distance between the objects, and the Hausdorff distance between the boundaries of the objects, should be small. According to the metric d_w, the ball with a thin hole in it is not close to the ordinary ball.

In the study of shape, however, it is often desired that for two objects to be considered close, they should have the same topological form. This also may have more than one interpretation: a stringent requirement is that two objects should be considered close only if they are linked by a homeomorphism that can be extended to all of R^3 or (slightly stronger) by an ambient isotopy [23, 32, 33]. This can be accomplished by adding a penalty function to d_w:

$$\delta(S, S') = \begin{cases} d_w(S, S') & \text{if } S \text{ and } S' \text{ are ambient isotopic} \\ \infty & \text{otherwise.} \end{cases}$$

(Informally, an ambient isotopy requires all of space to be transformed by a homeomorphism, which transforms the object along with the rest of space. To illustrate, a torus and a knotted torus are linked by a homeomorphism, but they are not linked by an ambient isotopy.)

Further discussion of metrics appropriate for comparing shapes can be found in [34], along with supporting theorems. Here we want only to emphasize two fundamental points. The first is the one already mentioned: if we want to judge the quality of a computed result, to be used by a method relevant to shape interrogation, then we must introduce a criterion to measure quality.

The second point relates to the idea of condition, which was introduced briefly in Section 4, and which will be discussed further in Section 6. Condition depends on the problem to be solved (for example, translation of two shapes may be well-conditioned, while regularized intersection of the same two shapes may be ill-conditioned), but it also depends crucially on the choice of metric. For example, an intersection problem involving proximate boundaries may be ill-conditioned with respect to a metric requiring sameness of topological form, but well-conditioned otherwise. The question of semantics appears again: to decide on the best choice of method, and on the best course of action, we must first specify what we *mean* by closeness of objects.

6 The semantics of ill-defined problems

In Section 4 it was observed that the data $\boldsymbol{g}$ in a representation $\Delta\boldsymbol{g}$ may not be known exactly: it may only be known, say, that $\boldsymbol{g}$ lies in a neighborhood of a given $\boldsymbol{g}_o$. Furthermore, the problem defined by $\boldsymbol{g}_o$ may be well or badly conditioned. Thus, in the example of Figure 3, the problem of regularized intersection is badly conditioned, relative to the metric d_w introduced in Section 5. Small changes in the position of the object S_1, for example, may cause parts of the boundary of $S_0 -^* S_1$ to simply disappear.

It is clear that in general there are two distinct sources of error involved: there

may be error due to methods, and there may be error due to the ill-condition of the problem itself. It turns out, however, that having error associated with the problem data may be turned to our advantage, in the sense that it may be possible to show that the additional error introduced by a numerical method has consequences no more serious than the consequences of error that must be tolerated in any event [35].

Returning to the second question posed at the beginning of Section 2, to ask a method to solve a problem that has not been completely specified, in the sense that the problem involves uncertain data, means that the method should find a "slightly wrong solution to a slightly wrong problem" [5]. This is in fact the definition of a *stable* method. The overall situation is illustrated in Figure 4, for the case of regularized Boolean intersection [3]. The arrows there correspond to the relation $\models$ discussed in Section 3. The inputs to the intersection algorithm are Δ_0 and Δ_1, and the computed output is Δ_c.

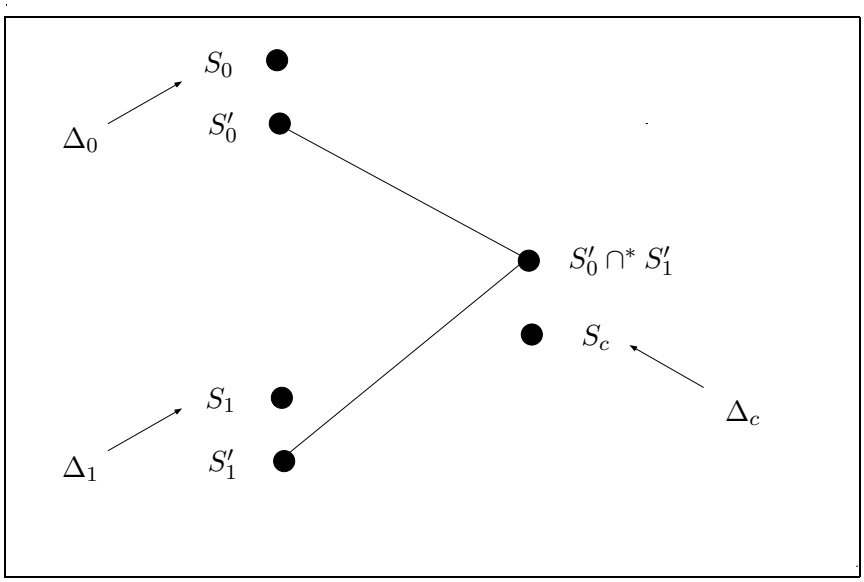

Figure 4: Error associated with problem and solution

The representations Δ_0, Δ_1 and Δ_c are the only things we can actually ob-

serve, but we can say something about the other items. First of all, if the representations are well-formed, then they correspond to actual subsets S_0, S_1 and S_c such that $S_0 \models \Delta_0$, $S_1 \models \Delta_1$ and $S_c \models \Delta_c$, with all that this implies (Section 3). An error analysis showing that a method is stable involves showing that there exists a problem defined by $[S_0', S_1']$, near to the problem defined by $[S_0, S_1]$, for which the true solution $S_0' \cap^* S_1'$ is near to S_c. The words "near to" are made precise by a metric, as described in Section 5. It seems apparent that such a proof must manipulate neighboring representations Δ_0', Δ_1' and $\Delta_{\cap^*}$ defining respectively the sets S_0' and S_1' such that $S_0' \models \Delta_0'$, $S_1' \models \Delta_1'$, and $S_0' \cap^* S_1' \models \Delta_{\cap^*}$. To give such a proof for a simple operation like set translation is quite simple; to give such a proof for the operation $\cap^*$ will certainly be quite difficult.

The first thing to notice is that a stable method does not necessarily provide us with a small error. In Figures 5 and 6, the space of problems is shown

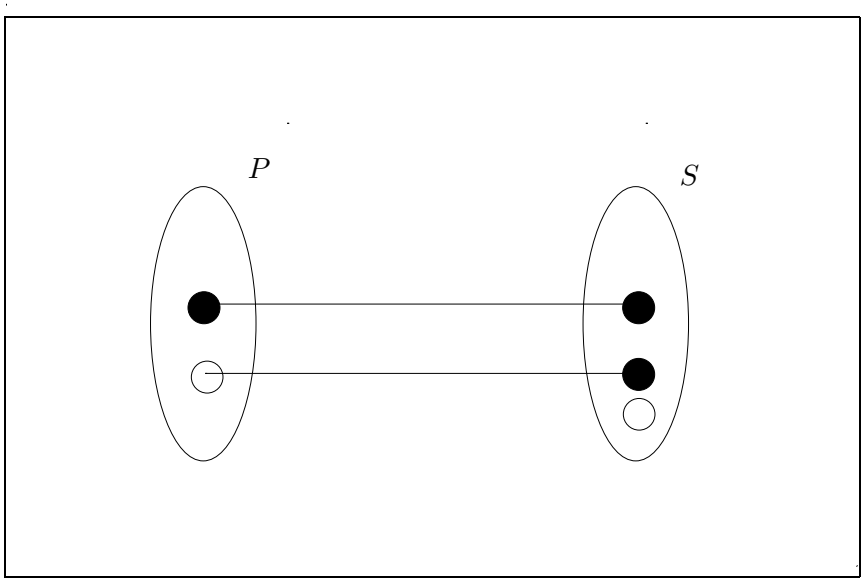

Figure 5: Well-conditioned case

on the left, and the space of solutions is shown on the right, with individual problems joined to their solutions by a line or curve. The problem presented to

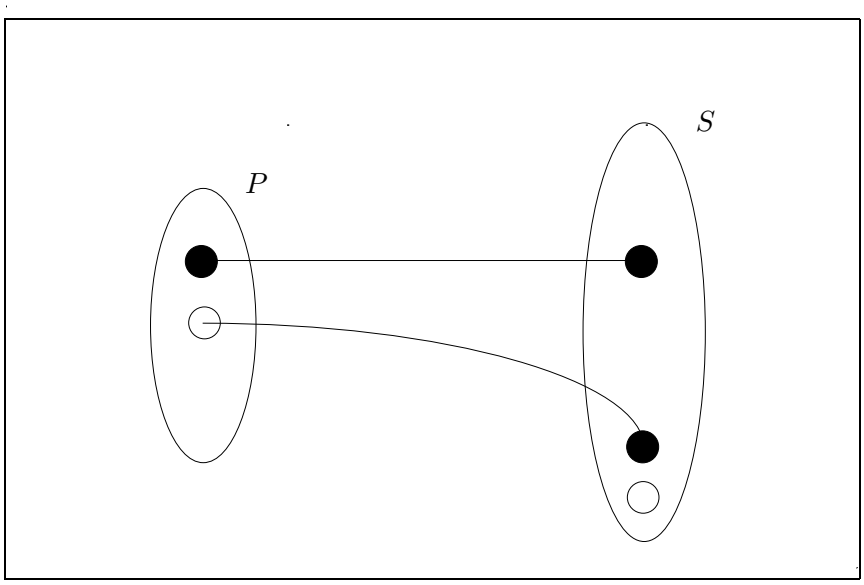

Figure 6: Ill-conditioned case

the method is shown by a black circle within the space of problems; the actual problem solved, and the actual solution obtained, are shown by unfilled circles. In the ill-conditioned case, the error in the solution provided by a stable method may be large.

The next thing to notice, however, is that if we are able to obtain a small error bound in solution space by associating part of the error with the problem, then in the case when there is uncertainty in the data, we have knowledge that we did not have before. This knowledge is, specifically, that if the perturbation of the problem introduced by the method is no larger than the uncertainty already present in the data, then the stable method will have provided a solution that is as good as the data warrants.

Rephrasing these ideas, the uncertainty in the data has provided us with the *luxury* of associating all or part of the data with the problem. However large the error is, the additional error introduced by a stable method is no worse than the error we must tolerate in any case.

Let us return now to the statements in Section 4, which said that if there is uncertainty in the given data, the use of exact arithmetic is not appropriate, and that we should not apply expensive exact arithmetic in a futile effort to obtain a satisfactory answer. This means, more precisely, that we should not use exact arithmetic to produce a solution more accurate than that produced by a stable method, since we cannot in any case avoid the consequences of uncertainty in the input data. This does not, however, rule out the possibility that exact arithmetic may be useful or necessary, in certain key parts of an algorithm, to produce a stable method in the first place. (This situation would be analogous to the case of solving the quadratic equation $ax^2 + bx + c = 0$: in order to get a satisfactory single-precision solution, double precision should be used in one key part of the algorithm, namely, the evaluation of the discriminant $b^2 - 4ac$ [5].)

Recently obtained perturbation results [36, 37] may be viewed as progress towards providing error analyses associating the error in surface intersection with the problem data.

It may be, of course, that we are not prepared to tolerate the consequences of a perturbation of the problem data, whatever its origin (data uncertainty, or contributed by a stable method). If the perturbation analysis (by which we mean [4, Ch. 2] the evaluation of problem condition) shows that the problem is ill-conditioned, then we may proceed to reformulate the problem, without even seeking a stable method. This was the case, for example, for the operation illustrated in Figure 3.

Yet another approach to dealing with ill-condition in the presence of uncertainty is to turn to methods outside the scope of numerical analysis. For example, in some situations it may be possible to modify the level of interaction with the user, so that ambiguous cases can be resolved by human intervention. Another possibility is to use heuristics, or other methods of artificial intelligence, to produce likely or plausible results, for example, by merging surfaces that seem to be similar. Such approaches may be quite reasonable, in some circumstances, but we should be conscious of what we are doing, and the user should be informed

about which type of method was used to obtain a given solution.

7 Consequences for implementation and for theory

The discussion above has direct practical consequences, some of which are immediate and relate to implementations, and some of which should affect the orientation of future theoretical development. We summarize some of the consequences here.

We begin with the question of implementations. First of all, when developing algorithms for shape interrogation in the presence of uncertainty, we should first specify what we mean by a good solution, and ask whether, in the light of given data and our choice of metric, a problem is ill-conditioned. If the answer is affirmative, then we should provide for the solution of a reformulated problem, for example, by requiring further information (as in the case of the example of Figure 3). That we are in such a situation is a valuable piece of information, and neither elaborate theorems nor elaborate implementations are required in this case: the appropriate next step is to reformulate the problem. It follows *a fortiori* that exact-arithmetic implementations are not appropriate in this case.

If, on the other hand, it is decided that for ill-conditioned problems furnishing a plausible solution is more important, or more practical, than guaranteeing absolute reliability, then development of algorithms can proceed using heuristic methods. That we are using a heuristic approach is also important information. No theorems are required, and development can proceed with the full knowledge that we are simply trying to produce an answer that is reasonable. Furthermore, in such a case, we can consider how to signal to the end user that the result is heuristic: a heuristic result, labelled as such, is much more valuable than a result whose reliability is uncertain.

A second immediate consequence for implementation is that it may be possible to provide certification of computed results if theorems like those in [22] are available, or optional certification when verification must be done computation-

ally, using algorithms such as those in [24].

Concerning the orientation of future theory, we first note that it would be useful to generalize results such as those in [22], which apply only to a special class of representations (curvilinear Bézier complexes), and to improve algorithms such as [24]. The discussion also shows the interest of generalizing results like those in [36] and [37], which permit association of error with the problem. Once this has been done, there will again be immediate consequences for implementation (for example, we should avoid the use of exact arithmetic if know already that we have the exact solution of a slightly perturbed problem).

Further consequences for future theory are that we should put high priority on finding metrics that better describe what mean by "close" in the context of solid modeling, and once we have done so, we should avoid formulating theorems giving error bounds that do not take problem condition into account. (Any such theorem will be unduly pessimistic in the case of well-conditioned problems.) Indeed, we should seek a stability result which guarantees that error is small if the problem is well-conditioned.

8 Conclusion

This paper gave answers to two fundamental semantical questions relevant in the context of shape interrogation.

9 Acknowledgments

The authors wish to thank G. Park and J.-R. Simard for their help and cooperation. They are also grateful to T. J. Peters, who made several useful comments. The second author is heavily indebted to his former teacher, W. M. Kahan. Responsibility for errors or omissions lies solely with the authors.

References

[1] Patrikalakis, N. and Maekawa, T. Shape Interrogation for Computer Aided Design and Manufacturing. Springer-Verlag, 2002.

[2] Farouki, R. Closing the gap between CAD Model and downstream application. *SIAM News* (32), No. 1, June 1999.

[3] Andersson, L.-E., Stewart, N. F. and Zidani, M. Error analysis for operations in solid modeling in the presence of uncertainty. Manuscript, February 13, 2004. Available at *www.iro.umontreal.ca/~stewart*.

[4] Wilkinson, The Algebraic Eigenvalue Problem. Oxford, 1965.

[5] Kahan, W. M. A survey of error analysis. Information Processing '71, North Holland, 1214-1239, 1971.

[6] Deuflhard, P. and Hohmann, A. Numerical Analysis in Modern Scientific Computation. An Introduction. Second Edition, Springer Verlag, New York, 2003.

[7] Requicha, A. A. G. Mathematical models of rigid solid objects. Technical Memo TM-28, University of Rochester, 1977.

[8] Fortune, S. Polyhedral modeling with exact arithmetic. *Third ACM Symposium on Solid Modeling*, Salt Lake City, 225-233, 1995.

[9] Keyser, J., Culver, T., Foskey, M., Krishnan, S. and Manocha, D. ESOLID: exact solid modeling for low-degree curved solids. *Seventh ACM Symposium on Solid Modeling*, 23-34, 2002.

[10] Spitz, S. and Rappoport, A. Integrated feature-based and geometric CAD data exchange. *Ninth ACM Symposium on Solid Modeling and Applications* (2004), G. Elber, N. Patrikalakis and P. Brunet (Editors), 183-190, Eurographics, 2004.

[11] Chen, X. and Hoffmann, C. Design compilation for feature-based, constraint-based CAD. *Third ACM Symposium on Solid Modeling*, Salt Lake City, 13-19, 1995.

[12] Chen, X. and Hoffmann, C. Towards feature attachment. *Computer-Aided Design* (27), 695-702, 1995.

[13] Chen, X. and Hoffmann, C. On editability of feature based design. *Computer-Aided Design* (27), 905-914, 1995.

[14] Mäntylä, M. Computational Topology: A study of topological manipulations and interrogations in computer graphics and geometric modeling. *Acta Polytechnica Scandinavica*, Mathematics and Computer Science Series No. 37, Helsinki, 1983.

[15] Lear, D. A. Practical applications of algebraic topology to geometric design. Working paper IFIP WG 5.2, Fourth IFIP WG 5.2 Workshop on Geometric Modeling and Computer-Aided Design, Rensselaerville, NY, Sept. 27 - Oct. 1, 1992.

[16] Piegl, L. and Tiller, W. The NURBS Book. Springer-Verlag, 1997.

[17] Brequet, G. and Sharir, M. Filling gaps in the boundary of a polyhedron. *Computer-Aided Geometric Design* (12), 207-229, 1995.

[18] Hoffmann, C., Park, G., Simard, J.-R. and Stewart, N. F. Residual iteration and accurate polynomial evaluation for shape-interrogation applications. *Ninth ACM Symposium on Solid Modeling and Applications* (2004), G. Elber, N. Patrikalakis and P. Brunet (Editors), Eurographics, 9-14, 2004.

[19] ACIS 3D Toolkit, Spatial Technology, Boulder, CO, 1998.

[20] STEP International Standard, ISO 10303-42, Industrial automation systems and integration — Product data representation and exchange — Part 42:

Integrated generic resources: Geometric and topological representation, International Organization for Standardization (ISO), Reference Number ISO 10303-42: 1994(E), First Edition 1994-12-15, 1997.

[21] Bamberg, P. and Sternberg, S. A Course in Mathematics for Students of Physics. Cambridge University Press, 1990.

[22] Andersson, L.-E., Peters, T. J. and Stewart, N. F. Selfintersection of composite curves and surfaces. *Computer Aided Geometric Design* (15), No. 5, 507-527, 1998.

[23] Andersson, L.-E., Peters, T. J. and Stewart, N. F. Equivalence of topological form for curvilinear geometric objects. *International J. of Computational Geometry and Applications* (10), No. 6, 609-622, 2000.

[24] Volino, P. and Thalmann, N. M. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Eurographics '94*, Vol. 13, No. 3, pp. C-155 to C-164. M. Daehlen and K. Kjelldahl, Guest Editors, Blackwell Publishers, 1994.

[25] Sakkalis, T., Shen, G. and Patrikalakis, N. M. Topological and geometric properties of interval solid models. *Graphical Models* (63), 163-175, 2001.

[26] Requicha, A. A. G. Toward a theory of geometric tolerancing. *Int. J. Robot. Res.* (2), No. 4, 45-60, 1983.

[27] Qi, J. and Shapiro, V. ϵ-solidity in geometric data translation. Technical Report SAL-2004-2, Spatial Automation Laboratory, University of Wisconsin-Madison, 2004.

[28] Hopcroft, J. E. and Kahn, P. J. A paradigm for robust geometric algorithms. Technical Report TR89-1044, Department of Computer Science, Cornell University, Ithaca, NY, 1989.

[29] Hoffmann, C. M., Hopcroft, J. E. and Karasick, M. S. Towards implementing robust geometric computations. Proceedings of the Fourth Annual ACM

Conference on Computational Geometry, Urbana-Champaign, IL, 106-117, 1988.

[30] Isaacson, E. and Keller, H. B. Analysis of Numerical Methods. Wiley, New York, 1966.

[31] Desaulniers, H. and Stewart, N. F. Backward error analysis for floating-point operations on rectilinear r-sets. TR #816, Département IRO, Université de Montréal, January 5, 1993.

[32] Boyer, M. and Stewart, N. F. Modeling spaces for toleranced objects. *Int. J. Robot. Res.* (10), No. 5, 570-582, 1991.

[33] Stewart, N. F. Sufficient condition for correct topological form in tolerance specification. *Computer-Aided Design* (25), No. 1, 39-48, 1993.

[34] Boyer, M. and Stewart, N. F. Imperfect-form tolerancing on manifold-objects: a metric approach. *Int. J. Robot. Res.* (11), No. 5, 482-490, 1992.

[35] Desaulniers, H. and Stewart, N. F. Robustness of numerical methods in geometric computation when problem data is uncertain. *Computer-Aided Design* (25), No. 9, 539-545, 1993.

[36] Song, X., Sederberg, T., Zheng, J., Farouki, R. T. and Hass, J. Linear perturbation methods for topologically consistent representations of free-form surface intersections. *Computer Aided Geometric Design* (21), 303-319, 2004.

[37] Farouki, R. T., Han, C. Y., Hass, J. and Sederberg, T. W. Topologically consistent trimmed surface approximations based on triangular patches. *Computer Aided Geometric Design* (21), 459-478, 2004.