Robustness of Boolean operations on subdivision-surface models

Di Jiang[1], Neil Stewart[2]

[1] Université de Montréal, Dép't. Informatique
CP6128, Succ. CentreVille, Montréal, H3C 3J7, Qc, Canada
jiangdi@umontreal.ca
[2] Université de Montréal, Dép't. Informatique
CP6128, Succ. CentreVille, Montréal, H3C 3J7, Qc, Canada
stewart@iro.umontreal.ca

Abstract. This work was presented in two parts at Dagstuhl seminar 08021. The two presentations described work in progress, including a "backward bound" for a combined backward/forward error analysis for the problem mentioned in the title.
We seek rigorous proofs that representations of computed sets, produced by algorithms to compute Boolean operations, are well formed, and that the algorithms are correct. Such proofs should eventually take account of the use of finite-precision arithmetic, although the proofs presented here do not.
The representations studied are based on subdivision surfaces. Such representations are being used more and more frequently in place of trimmed NURBS representations, and the robustness analysis for these new representations is simpler than for trimmed NURBS.
The particular subdivision-surface representation used is based on the Loop subdivision scheme. The analysis is broken into three parts. First, it is established that the input operands are well-formed two-dimensional manifolds without boundary. This can be done with existing methods. Secondly, we introduce the so-called "limit mesh", and view the limit meshes corresponding to the input sets as defining an approximate problem in the sense of a backward error analysis. The presentations mentioned above described a proof of the corresponding error bound. The third part of the analysis corresponds to the "forward bound": this remains to be done.

Keywords. Robustness, finite-precision arithmetic, Boolean operations, subdivision surfaces

1 Introduction

The problem considered here forms part of the "robustness problem" in solid modelling. We seek rigorous proofs that representations are well formed, and that algorithms operating on these representations are correct.

Dagstuhl Seminar Proceedings 08021
Numerical Validation in Current Hardware Architectures
http://drops.dagstuhl.de/opus/volltexte/2008/1443

The robustness problem has received a great deal of attention over the last two decades. Much of the work focused on representations based on trimmed-NURBS representations, or on simplified versions of this representation based on planar faces. The trimmed-NURBS representation has been quite carefully defined by *de facto* industrial standards [1], and by international standards [2].

There are serious difficulties involved in providing rigorous bounds in the case of trimmed-NURBS representations. These difficulties arise from the use of low-order polynomial approximations for inter-patch boundaries (in both the parametric and object-space domains), as well as from the use of finite-precision arithmetic. A fundamental fact to be noted is that errors introduced from these sources may lead to representations that are not well-formed. This means that it is not a matter of small errors leading to representations that are more or less in error, depending, say, on the condition of the problem to be solved. Rather, small errors may lead to representations that do not define any subset of $\mathbb{R}^3$. These difficulties are described in [3] and [4], and references to the literature on the subject are given there.

In recent years, the subdivision-surface approach [5] has gained popularity as a representation method for solids in $\mathbb{R}^3$. One of the main reasons for this is that the problems, mentioned above, of matching boundaries, are avoided. Corresponding to this fact is the possibility of giving rigorous proofs for standard algorithms, such as Boolean intersection of objects defined by the subdivision-surface representation. Such proofs are very difficult in the trimmed-NURBS case [4]. In short, we are concerned here with an easier case (subdivision-surface models) than the trimmed-NURBS case, but this easier case corresponds to what is likely to be the primary representation method used in the medium-term future.

This document corresponds to two presentations, given by the authors at Dagstuhl Seminar 08021, January 6-11, 2008. These presentations described work in progress, as explained below. Detailed proofs will be given in the first author's PhD thesis [6] and in a subsequent paper.

2 Problem outline

Subdivision surfaces may be viewed as generalizations of uniform tensor-product B-splines. The generalization may occur in two separate directions.

First of all, uniform tensor-product B-splines are defined on the doubly bi-infinite two-dimensional grid $\mathbb{Z}^2$, and the corresponding subdivision algorithms provide, as they proceed, a progressively better approximation to the B-spline surface. In contrast, subdivision-surface algorithms work in an arbitrary locally-planar mesh, which means that the boundaries of the objects concerned may have the generality of an arbitrary two-manifold (in fact, it is even possible to define subdivision algorithms generating so-called "non-manifold" objects). Furthermore, variants of these subdivision algorithms, such as the Catmull-Clark and Doo-Sabin algorithms, permit non-regularity in the locally-planar mesh, *i.e.*, the meshes may have topology different from regular planar tilings.

Secondly, many subdivision algorithms, including the Loop method, which is studied in this paper, are based on splines that are more general than the tensor-product B-splines. Some, such as the Loop method, are based on box splines, which include the tensor-product B-splines as a special case. Others, such as the $\sqrt{3}$ method and the Kobbelt method, are based on an even more general class of splines.

The locally-planar mesh defining a subdivision surface is a logical structure, with control points attached to the logical vertices of the mesh. The control points are in $\mathbb{R}^N$, where often $N = 3$. The surfaces generated by the subdivision algorithms mentioned above can be manipulated to change their position and orientation, and they can be generalized to include so-called *hard* or *crease* edges, for the purpose of various modelling operations. Further, classical solid-modelling operations such as regularized Boolean intersection [7] are also of interest, and algorithms to realize such operations have been presented in the literature [8].

One interesting fact about subdivision surfaces generated by standard methods, such as the Loop method, is that it is possible to "push" any given control point to the limit, *i.e.*, to find the point in the surface to which the given control point will converge as a result of the subdivision process. In [6] a new algorithm for Boolean operations will be presented, which is based on the use of the limit mesh obtained by pushing control points to their limit. In this paper we discuss certain bounds on the position of the corresponding surfaces, to be used in showing that the algorithm is stable.

The remainder of the paper is organized as follows. In the next section we give an outline of the classical backward error analysis, as it applies in the context of solid modelling. Then, in Section 4, the derivation and statement, of the bound described in the previous paragraph, are presented. Section 5 concludes the paper.

3 Backward error analysis

The backward error analysis in numerical analysis has as goal a proof that a method has found approximately the right solution to approximately the right problem [3,4,9]. The difference between this kind of analysis, and a simple "forward" analysis, is that all or part of the error is associated with the problem to be solved. This is very convenient in a context where there is uncertainty in the input data: for example, we may be able to show that the algorithm finds a solution to a problem that is no further from the input problem than the uncertainty. In this case, the algorithm has given us an answer that is as good as we can hope for. It does not mean that the error will be small: if the problem is ill-conditioned, the error will be large even though we have solved a problem that is close to the problem presented to the method. But if the problem is ill-conditioned, we must live with a large error in any case, due to the effect of the uncertainty in the input data.

In our context, the following is an outline of the backward error analysis for the problem of computing the regularized Boolean intersection $S_1 \cap^* S_2$ of

two solids S_1 and S_2. Each of these solids S_i is a three-dimensional manifold with boundary, each is a subset of $\mathbb{R}^3$, and each is defined by its boundary $\partial S_i, i = 1, 2$. The boundary ∂S_i is implicitly defined by the following pair: a locally-planar triangular mesh $\check{M}$, and the Loop subdivision method.

Outline of backward error analysis in Boolean-operation case

The outline is as follows.

1. Establish *a priori* or *a posteriori* that the input arguments ∂S_1 and ∂S_2 are well-formed two-dimensional manifolds without boundary.
2. Introduce a mesh $\bar{M}_i$ that is topologically the same (homeomorphic) as the input argument $\partial S_i, i = 1, 2$. View the $\bar{M}_i$ as defining the approximate problem, actually solved by the numerical method, in the sense of the backward error analysis.
3. Bound the difference between ∂S_i and $\bar{M}_i$ on a face-by-face basis.

We may then write $\partial S_i \cong \bar{M}_i$, where this means that the boundaries ∂S_i of the input sets are geometrically close to the sets $\bar{M}_i$, and that they are topologically the same. We would then like to prove that the algorithm finds the exact intersection of the approximate problem of computing the regularized intersection of the two sets having boundaries $\bar{M}_1$ and $\bar{M}_2$, respectively. This would be a pure backward analysis. If only a weaker theorem is possible, so that the word "exact" in the last sentence is replaced by an analysis involving some error, then the analysis is a combined forward-backward analysis [9].

The first item in the list can be accomplished by means of methods like that of Volino and Thalmann [10,11], to detect possible illegal intersections between adjacent faces of the solids. This approach has been used before in the context of subdivision-surface models by, for example Grinspun and Schröder [12]. Methods like those of Kobbelt [13], or Wu and Peters [14,15], can be used to detect the possible illegal intersection of faces that are supposed to be disjoint.

For the second item in the list, for each of the two input arguments $\partial S_i, i = 1, 2$, we introduce a mesh obtained by pushing all of the objects' control points to their respective limits. It is this that we call the *limit mesh* $\bar{M}_i$, and we view the surface obtained by linearly interpolating between the limit points as the surface of the set defining an input to the approximate problem.

The third item in the list, bounding the difference between ∂S_i and $\bar{M}_i$, is discussed in the next section.

Preliminary discussion

There are two contributions envisaged in this paper. One is to suggest an algorithm for Boolean intersection that will be more accurate and more efficient, since the limit mesh is a better approximation to the boundary of the solid than the original control mesh. The second is to give at least part of a backward error analysis, as described above. This does not mean, however, that both contributions will be simultaneously useful for any particular example.

The bounds on the error in mesh approximations that can be computed at reasonable computational cost are not small, unless subdivision has proceeded nearly to convergence. This is true of the bounds that have appeared previously in the literature, and it is true of the bounds described here. Suppose first that we are dealing with the case when few iterations have been effected. It is in this case that use of the limit mesh, as opposed to the original control mesh, will give an improved result, in the sense of a forward error analysis. Unless the uncertainty in the input data is very large, however, this will not necessarily be reflected in the backward error analysis. If the uncertainty in the input data is small, say on the order of a pixel width, the backward error analysis will be unable to affirm that the method has given a satisfactory result. Although perhaps disappointing, this is worthwhile information in itself: the purpose of an error analysis is not to show that all methods give satisfactory results, but, rather, to describe under what conditions the method gives satisfactory results.

On the other hand, if the method has been iterated close to convergence, the backward error analysis will be able to affirm that a satisfactory result has been obtained, whether or not the limit mesh has been used. Again, this is useful information, even if using the limit mesh provides no special advantage in this case.

4 Bounding the facewise difference between ∂S_i and $\bar{M}_i$

The subdivision algorithm studied is the Loop method, with a parametrization suggested in [14]. In this section a bound on the facewise difference between ∂S_i and $\bar{M}_i$ is given. Improved bounds will be given in [6].

An extraordinary vertex v_0 (valence $n \neq 6$) is set to be the origin in the u-v domain, its one-ring neighbors form a unit n-gon, and Ω_n represents the domain of the triangular patch corresponding to the triangle 0-1-2 (Figure 1, left).

4.1 Representations

To simplify the notation, we omit the subscript i on ∂S_i. At subdivision level ι, we distinguish amongst three different representations: the *control mesh* $\breve{M}^\iota$, the *limit mesh* $\bar{M}^\iota$ and the *limit surface* ∂S. Now, having introduced the index ι in order to be precise, we immediately suppress it in order to simplify the notation.

Control mesh $\breve{M}$

Each face $\breve{\boldsymbol{F}}$ of the *control mesh* $\breve{M}$ of a Loop subdivision-surface model can be evaluated as

$$\breve{\boldsymbol{F}}(u,v) = \sum_{i=0}^{2} \boldsymbol{p}_i \cdot \bar{b}_i(u,v) \tag{1}$$

where $\boldsymbol{p}_i,\ i = 0, 1, 2$ are the control points and $\bar{b}_i$ is a function defining piecewise-linear interpolation between the control points: $\bar{b}_i(u,v)$ is equal to one at vertex i, equal to zero at vertices that are immediate neighbours of i, linear on the polygon centered at i, and zero elsewhere (see Figure 1, right).

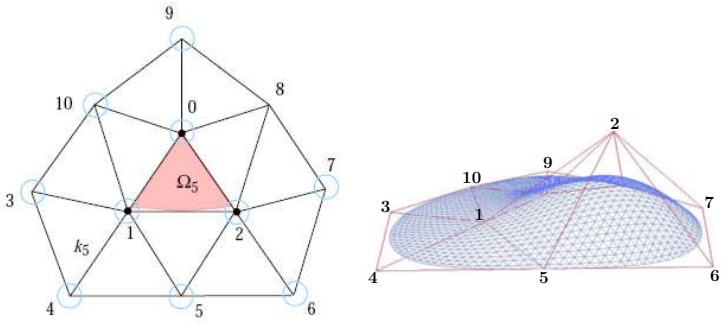

Fig. 1. Left: a base mesh used to generate the basis functions for the triangle 0-1-2 (irregular case: vertex with valence $n = 5$) [14]; right: the resulting basis function b_i at node $i = 2$ evaluated at level four, and $\bar{b}_i$.

Limit mesh $\bar{M}$

The *limit mesh* $\bar{M}$ of a subdivision-surface model is formed by pushing the vertices of the *control mesh* $\breve{M}$ to their corresponding limit positions. Under Loop subdivision, each face $\bar{\boldsymbol{F}}$ of $\bar{M}$ can be evaluated as

$$\bar{\boldsymbol{F}}(u,v) = \sum_{i=0}^{2} \bar{\boldsymbol{p}}_i \cdot \bar{b}_i(u,v) \tag{2}$$

where $\bar{\boldsymbol{p}}_i$ are the limit position points of the control points $\boldsymbol{p}_i$, $i = 0, 1, 2$ which can be computed by applying a limit position matrix [16,17] to the original control points $\boldsymbol{p}_i$.

Limit surface ∂S

A *limit patch* $\boldsymbol{F}$ of the *limit surface* ∂S under Loop subdivision can be evaluated as a linear combination of basis functions with the original control points as weights

$$\boldsymbol{F}(u,v) = \sum_{i=0}^{n+5} \boldsymbol{p}_i \cdot b_i(u,v) \tag{3}$$

where $\boldsymbol{p}_i, i = 0, \ldots, n+5$ are the control points (with (x, y, z) coordinates) of the *control mesh* $\breve{M}$, n is the valence of the control point corresponding to $i = 0$ (see Figure 1) of the limit patch in question, and b_i is the function, centered at vertex i embedded in the u-v plane, defined by applying Loop subdivision to the unit impulse centered in the i-th vertex.

As in [14,15], we can extract a linear function $\boldsymbol{\ell}$ interpolating the three vertices $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2$ of the central triangle, and rewrite the limit patch $\boldsymbol{F}(u, v)$ as:

$$\boldsymbol{F}(u, v) = \boldsymbol{\ell}(u, v) + \sum_{i=3}^{n+5} \boldsymbol{d}_i \cdot b_i(u, v). \tag{4}$$

Here, $\boldsymbol{d}_i$ is the vector whose components correspond to the difference between $\boldsymbol{\ell}(u_i, v_i)$ and $\boldsymbol{p}_i$ along each axis as

$$\boldsymbol{d}_i = \boldsymbol{p}_i - \boldsymbol{\ell}(u_i, v_i), \;\; i = 0, \ldots, n+5, \tag{5}$$

and for $i = 0, 1, 2$, $\boldsymbol{d}_i = 0$ by definition. (Our notation differs slightly from [14,15]: in particular, we use boldface to denote elements of $\mathbb{R}^3$.) Then,

$$\begin{aligned} \boldsymbol{\ell}(u, v) + \sum_{i=3}^{n+5} \boldsymbol{d}_i b_i(u, v) &= \boldsymbol{\ell}(u, v) + \sum_{i=3}^{n+5} [\boldsymbol{p}_i - \boldsymbol{\ell}(u_i, v_i)] b_i(u, v) \\ &= \boldsymbol{F}(u, v) + \{\boldsymbol{\ell}(u, v) - \sum_{i=0}^{n+5} \boldsymbol{\ell}(u, v) b_i(u, v)\} \\ &\equiv \boldsymbol{F}(u, v) \end{aligned}$$

since Loop subdivision scheme reproduces linear functions.

For each vertex $\boldsymbol{v}_i$, $i = 3, \ldots, n+5$, we can find its barycentric coordinates $(s_i, t_i, 1 - s_i - t_i)$ with respect to the three vertices $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2$ of the central triangle in the parametric domain (u-v domain), *i.e.*

$$\boldsymbol{v}_i = s_i \cdot \boldsymbol{v}_0 + t_i \cdot \boldsymbol{v}_1 + (1 - s_i - t_i) \cdot \boldsymbol{v}_2, \;\; i = 3, \ldots, n+5.$$

Then the linear function $\boldsymbol{\ell}$ can be expressed as

$$\boldsymbol{\ell}(u, v) = s \cdot \boldsymbol{p}_0 + t \cdot \boldsymbol{p}_1 + (1 - s - t) \cdot \boldsymbol{p}_2, \quad u, v \in R^2. \tag{6}$$

and therefore

$$\boldsymbol{\ell}(u_i, v_i) = s_i \cdot \boldsymbol{p}_0 + t_i \cdot \boldsymbol{p}_1 + (1 - s_i - t_i) \cdot \boldsymbol{p}_2 \;\; i = 0, \ldots, n+5. \tag{7}$$

Thus, the difference $\boldsymbol{d}_i$ can be computed as:

$$\boldsymbol{d}_i = \boldsymbol{p}_i - (s_i \cdot \boldsymbol{p}_0 + t_i \cdot \boldsymbol{p}_1 + (1 - s_i - t_i) \cdot \boldsymbol{p}_2), \;\; i = 0, \ldots, n+5, \tag{8}$$

where $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ correspond to the vertices of the central triangle in the control mesh $\breve{M}$.

4.2 Error bound

Due to the application of the Loop subdivision to the parametric plane, the domain Ω as defined in [14] of the limit patch $\boldsymbol{F}$ is the limit of the subdivision

applied to the initial mesh of the parameteric u-v domain. Similarly, the domain $\bar{\Omega}$ of its corresponding face $\bar{\boldsymbol{F}}$ of the limit mesh $\bar{M}$ is a triangle in the initial mesh of the parametric u-v domain itself. We define a mapping function $\phi : (u, v) \rightarrow (u', v')$, where $(u', v') \in \Omega$ and $(u, v) \in \bar{\Omega}$; then, the difference between the limit surface ∂S and the surface corresponding to the limit mesh $\bar{M}$ can be defined as:

$$||\boldsymbol{F}((\phi(u, v)) - \bar{\boldsymbol{F}}(u, v))||, \quad (u, v) \in \bar{\Omega}.$$

Let $\mathcal{N}_i$ be the indices of points in the one-ring neighborhood of control point $\boldsymbol{p}_i$, and $\mathcal{L}_i$ denote the *Laplacian coordinate* of $\boldsymbol{p}_i$, defined as:

$$\mathcal{L}_i = \boldsymbol{p}_l - \frac{1}{n_i} \sum_{j \in \mathcal{N}_i} \boldsymbol{p}_j$$

where $i \in \mathcal{I}$, the index set of mesh vertices, n_i is the valence of vertex $\boldsymbol{p}_i$, and each $\boldsymbol{p}_j$ is a one-ring neighbor of vertex $\boldsymbol{p}_i$. In the proof below we have $i = 0, 1, 2$ and $n_0 = n, n_1 = n_2 = 6$.

Lemma 1. *For $(u', v') \in \Omega$, $(u, v) \in \bar{\Omega}$ and $(u', v') = \phi(u, v)$,*

$$||\boldsymbol{F}(u', v') - \bar{\boldsymbol{F}}(u, v)|| = ||\sum_{i=0}^{2} \alpha_i \cdot \bar{b}_i(u, v) \cdot n_i \cdot \mathcal{L}_i + \sum_{i=3}^{n+5} \boldsymbol{d}_i \cdot b_i(\phi(u, v))|| \tag{9}$$

Proof.

$$\begin{aligned}
||\boldsymbol{F}(u', v') - \bar{\boldsymbol{F}}(u, v)|| &= ||\boldsymbol{F}(\phi(u, v) - \bar{\boldsymbol{F}}(u, v)|| \\
&= ||\sum_{i=0}^{n+5} \boldsymbol{p}_i \cdot b_i(\phi(u, v)) - \sum_{i=0}^{2} \bar{\boldsymbol{p}}_i \cdot \bar{b}_i(u, v)|| \\
&= ||\sum_{i=0}^{2} \boldsymbol{p}_i \cdot \bar{b}_i(u, v) + \sum_{i=3}^{n+5} \boldsymbol{d}_i \cdot b_i(\phi(u, v)) - \sum_{i=0}^{2} \bar{\boldsymbol{p}}_i \cdot \bar{b}_i(u, v)|| \\
&= ||\sum_{i=0}^{2} (\boldsymbol{p}_i - \bar{\boldsymbol{p}}_i) \cdot \bar{b}_i(u, v) + \sum_{i=3}^{n+5} \boldsymbol{d}_i \cdot b_i(\phi(u, v))||
\end{aligned}$$

With the limit position mask we have

$$\begin{aligned}
\boldsymbol{p}_i - \bar{\boldsymbol{p}}_i &= \boldsymbol{p}_i - [(1 - n_i \cdot \alpha_i) \cdot \boldsymbol{p}_i + \alpha_i \cdot \sum_{j \in \mathcal{N}_i} \boldsymbol{p}_j] \\
&= \boldsymbol{p}_i - (1 - n_i \cdot \alpha_i) \cdot \boldsymbol{p}_i - \alpha_i \cdot \sum_{j \in \mathcal{N}_i} \boldsymbol{p}_j \\
&= n_i \cdot \alpha_i \cdot \boldsymbol{p}_i - \alpha_i \cdot \sum_{j \in \mathcal{N}_i} \boldsymbol{p}_j \\
&= n_i \cdot \alpha_i \cdot (\boldsymbol{p}_i - \frac{1}{n_i} \cdot \sum_{j \in \mathcal{N}_i} \boldsymbol{p}_j) \\
&= n_i \cdot \alpha_i \cdot \mathcal{L}_i, i = 0, 1, 2,
\end{aligned}$$

where $\mathcal{N}_i$ is the one-ring neighborhood of control point $\boldsymbol{p}_i$. Then we have

$$\begin{aligned}\|\boldsymbol{F}(u',v') - \bar{\boldsymbol{F}}(u,v)\| &= \|\sum_{i=0}^{2}(\boldsymbol{p}_i - \bar{\boldsymbol{p}}_i)\cdot \bar{b}_i(u,v) + \sum_{i=3}^{n+5}\boldsymbol{d}_i \cdot b_i(\phi(u,v))\| \\ &= \|\sum_{i=0}^{2}\alpha_i \cdot \bar{b}_i(u,v)\cdot n_i \cdot \mathcal{L}_i + \sum_{i=3}^{n+5}\boldsymbol{d}_i \cdot b_i(\phi(u,v))\|\end{aligned}$$

Definition 1. *The weighted Laplacian coordinate can be denoted as*

$$\tilde{\mathcal{L}}_i = \boldsymbol{p}_i - \frac{1}{\mu_i}\cdot\sum_{j\in\mathcal{N}_i}\omega_j\cdot\boldsymbol{p}_j \tag{10}$$

with $\mu_i = \sum_{j\in\mathcal{N}_i}\omega_j$, *and* $i \in \mathcal{I}$.

Note that the ω_j may also depend on i. If $\frac{\omega_j}{\mu_i} = 1/n_i$, then $\tilde{\mathcal{L}}_i = \mathcal{L}_i$.

With the above definition for the weighted Laplacian operator, we have

Lemma 2.

$$\sum_{i=3}^{n+5}\boldsymbol{d}_i \cdot b_i(\phi(u,v)) = -\sum_{i=0}^{2}\mu_i\cdot\tilde{\mathcal{L}}_i \tag{11}$$

where the weights u_i *are available from [6].*

The proof of this Lemma, and of the subsequent theorem, are omitted here.

Let $\boldsymbol{F}(u,v)$ denote the *limit surface* face and let $\bar{\boldsymbol{F}}(u,v)$ represent the *limit mesh* face. We have

Theorem 1. *For* $(u',v') \in \Omega$, $(u,v) \in \bar{\Omega}$ *and* $(u',v') = \phi(u,v)$, *we have*

$$\begin{aligned}\|\boldsymbol{F}(u',v') - \bar{\boldsymbol{F}}(u,v)\| &= \|\sum_{i=0}^{2}\alpha_i\cdot\bar{b}_i(u,v)\cdot n_i\cdot\mathcal{L}_i - \mu_i\cdot\tilde{\mathcal{L}}_i\| \\ &\le \|\sum_{i=0}^{2}\alpha_i\cdot\bar{b}_i(u,v)\cdot n_i\cdot\mathcal{L}_i\| + \|\sum_{i=0}^{2}\mu_i\tilde{\mathcal{L}}_i\|.\end{aligned}$$

There is normally some cancellation in the difference in the righthand side of the equality in the theorem, and the bound can be modified to take account of this. Improved bounds will be given in [6].

5 Conclusion

In this paper we have described how an intersection algorithm, based on the use of the limit mesh obtained by pushing control points to their respective limits, can be used to provide more accurate intersections in the sense of a forward error analysis. We have also given a preliminary bound that could be used as the basis of a backward error analysis for this algorithm; improved bounds, however, will be given in [6].

References

1. ACIS 3D Toolkit, Spatial Technology, Boulder, CO, 1998.
2. STEP International Standard, ISO 10303-42, Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 42, International Organization for Standardization (ISO), Reference Number ISO 10303-42: 1994 (E), First Edition 1994-12-15, 1997.
3. Hoffmann, C. M., and Stewart, N. F. Accuracy and semantics in shape-interrogation applications. *Graphical Models* (67), No. 5, 373-389, September 2005.
4. Andersson, L.-E., Stewart, N. F. and Zidani, M. Error analysis for operations in solid modeling in the presence of uncertainty. *SIAM J. Sci. Comput.* (29), No. 2, 811-826, 2007.
5. Zorin, D. and Schröder, P. Subdivision for Modeling and Animation. SIGGRAPH 2000 Course Notes, 2000.
6. Jiang, D. PhD Thesis, Université de Montréal, Département IRO, 2008.
7. Requicha, A. A. G. Representations for rigid solids: theory, methods and systems. *Computing Surveys* (12), No. 4, 437-464, 1980.
8. Biermann, H., Kristjansson, D. and Zorin, D. Approximate Boolean operations on free-form solids. *SIGGRAPH'01*, 185-194, 2001.
9. Kahan, W. M. A survey of error analysis. International Federation of Information Processing: IFIP71, North-Holland, 1214-1239, 1972.
10. Volino, P. and Thalmann, N. M. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In: Daehlen, M., Kjelldahl, K. (eds), *Eurographics '94*, (13), No. 3, C-155-C-164, 1994.
11. Andersson, L.-E., Stewart, N. F., and Zidani, M. Conditions for use of a non-selfintersection conjecture. *Computer Aided Geometric Design* (23), 599-611, 2006.
12. Grinspun, E. and Schröder, P. Normal bounds for subdivision-surface intersection. *Proceedings of the IEEE Conference on Visualization*, 333-340, 2001.
13. Kobbelt, L. Tight bounding volumes for subdivision surfaces. *Pacific Graphics '98*, 17-26, 1998.
14. Wu, X., and Peters, J. Interference detection for subdivision surfaces. *Eurographics 2004* (23), No. 3, 577-585, 2004.
15. Wu, X., and Peters, J. An accurate error measure for adaptive subdivision surfaces. *Shape Modeling International (SMI2005)*, 2005.
16. Hoppe, H. *et al*. Piecewise smooth surface reconstruction. *SIGGRAPH'94*, 295-302, 1994.
17. Wa, M. *et al*. A direct approach for subdivision surface fitting from a dense triangle mesh. *Computer-Aided Design* (36), No. 6, 525-536, 2004.