Computer-Aided Design and Applications
© 2008 CAD Solutions, LLC
http://www.cadanda.com

Boolean Set Operations with Cubic Algebraic Patches

Chandrajit Bajaj[1], Alberto Paoluzzi[2], Simone Portuesi[3], Na Lei[4] and Wenqi Zhao[5]

[1]Univ. of Texas at Austin, USA, bajaj@cs.utexas.edu
[2]Univ. "Roma Tre", Italy, paoluzzi@dia.uniroma3.it
[3]Univ. "Roma Tre", Italy, portuesi@dia.uniroma3.it
[4]Jilin University, China, leina@jlu.edu.cn
[5]Univ. of Texas at Austin, USA, wzhao@ices.utexas.edu

ABSTRACT

We present a symbolic-numeric algorithm for Boolean set operations, closed in the algebra of curved polyhedra triangulated with compactly supported cubic algebraic surface patches. The input simplicial polyhedral representation includes the vertices of the triangulation and prescribed vertex normals. On each triangle of this boundary representation, a piecewise cubic algebraic interpolant is constructed, using a C^1-continuous algebraic patch (prism A-patch) that interpolates the triangle vertices, with given normals. In order to represent flat and/or sharp local features, the normal-per-face and/or normal-per-edge may be given, respectively. The boundary topology is described by storing, for each triangle, the two triples of pointers to incident vertices and to adjacent triangles. For each triangle, a scaffolding prism is built, which provides a containment volume for the interpolating A-patch. When looking for the result of a regularized Boolean operation, the 0-set of a three-variate polynomial within each such prism is generated, and intersected with the 0-sets of the other curved polyhedron, when two prisms have non-empty intersection. The intersection curves are traced and used to decompose each boundary into the 3 standard classes of subpatches, denoted *in*, *out* and *on*. While tracing the intersection curves, a locally refined triangulation of intersecting patches is produced, and added to the boundary representation.

Keywords: Boolean operations, A-patches, Free-form surface, Boundary representation.
DOI: 10.3722/cadaps.2008.730-742

1. INTRODUCTION

A-Patches are smooth algebraic surface patch families, defined using a fixed degree trivariate polynomial within a compact polyhedron domain (also called the patch scaffold). Simple A-patch elements use a tetrahedron, or a cube, or a triangular prism scaffold. An exact union or intersection Boolean operator is often not closed inside the domain of fixed degree A-patches. The reason is simply that the intersection of two algebraic surfaces of degree d (say cubic) is in general, a space curve of degree d^2, (i.e. nine). In this paper, we provide an efficient solution, using a clever combination of symbolic and geometric methods, for the subproblems below:

(i) The robust computation of the intersection of a pair of algebraic curves, and/or surfaces;
(ii) The decomposition of the explicit union of a pair of A-patches, into a small set of A-patches of the same scaffold type.

The combination of (i) and (ii) provides a union operator capable of reproducing A-patches maintaining the topology of the exact solution. These methods are currently being implemented in the context of the PLaSM Language and using the Ganith Algebraic Toolkit. The computation of the union of A-patches is a necessary step for the development of Boolean set operations on algebraic finite elements in an integrated software package.

The main assumption we make here is that the boundary of the solid is triangulated by curved algebraic patches, where each patch is single-sheeted and contained within the prismatic scaffold on the triangles of the coarse linear approximation of the solid boundary. This assumption reduces the Boolean problem with curved solids to the much easier and well-known problem with linear polyhedra (see, e.g. [3], [12],[16], [17], [18]). For the same reason, we may represent quite complex curved solids using the very simple winged-triangle (WT) boundary representation, which does not require the use of Euler operators [8]. Such winged representation was introduced in [21], and later generalized to solid decompositions and higher dimensional manifolds in [20].

Previous and related work Modeling of solids with implicit algebraic patches, include C^1 piecewise quadric patches [7]. Clough-Tocher split for C^1 cubic patches [11]. Single valued cubic C^1 A-patches [1]. Quintic C^2 A-patches [6]. Rational C^1 A-patches [22]. C^1 Prism A-patches and shell A-patches [4], [5]. Boolean/Trimming of free form surfaces include: Interactive boundary computation of Boolean combinations of sculptured solids [14]. Netbased modeling [15]. Approximate Boolean operations on free-form solids [2]. ESOLID - a system for exact boundary evaluation [13]. Adaptive trimming of cubic triangular Bézier patches [9].

The rest of this paper is organized as follows. In Section 2 some useful concepts are recalled and preliminary definitions are given. In particular, we discuss in detail in Sections 2.2.1 the mathematical toolbox required for manipulating triangular (and quadrangular) prismatic A-patches. In Section 3 we give a synthetic introduction to Boolean algorithms between curved triangulated solids. In Section 4 we discuss the implemented intersection between prismatic A-patches, with tracing of intersection curves and local triangulation refinement. In the Examples Section 5 we show the results of our prototype implementation, including the computation of the Boolean union of the docked complex of two proteins containing several hundreds of cubic patches. In the Conclusion section 6 the open problems are described and future extensions and work are outlined.

2. BACKGROUND

2.1 Preliminaries

Polyhedron An *m-polyhedron* [22] is any compact set $P \subset \mathbb{E}^n$ that allows at least one triangulation (K,h), where K is a simplicial *m-complex*, and $h : |K| \to P$ a homeomorphism, where the support space of K, denoted as $|K|$, is the point-set union of simplices $\sigma \in K$. An *m-polyhedron* is said *linear* if h is the identity function; it is said *regular* if the complex K associated with it is pure, i.e. if each simplex is a face of some *m-simplex*. Two simplices σ_1 and σ_2 in a complex K are *s-adjacent* if they have a common *s-face*; they are *s-connected* if a sequence of simplices in K exists, beginning with σ_1 and ending with σ_2, such that any two successive terms of the sequence are *s-adjacent*.

Surface We call *surface* any 2-polyhedron which supports a triangulation (K,h) such that:

(i) any 1-simplex in K is a face of at most two 2-simplices in K;

(ii) for each 0-simplex $v \in K$, the 2-simplicies of K for which v is a face can be circularly ordered so that any consecutive pair is 1-adjacent.

Such conditions impose that a surface should be a *manifold* object, where the neighbourhood of each point is homeomorphic to the open disc. If the first condition is strictly verified, i.e. if each 1-simplex in K is a face of exactly two 2-simplices in K, then the surface is *closed*; otherwise it is *open*. If the surface is closed and orientable, then it is the boundary of a solid.

Shell A maximal 1-connected component of a closed surface is called a *shell* of the surface.

Boolean algebra The domain of the representation scheme [23] discussed in this paper is the Boolean algebra of solid polyhedra with bounded boundary and where complementation simply exchanges the interior with the exterior. It is easy to see that it verifies all axioms of a Boolean algebra, and that it also has to verify all theorems known for a Boolean algebra. In particular, we may evaluate the intersection and difference of polyhedra from their union and complements, using the notion that conjunction is dual to disjunction by way of the De Morgan's laws. To complement any boundary representation of a solid is $O(n)$, since it is sufficient to reverse the direction of all the normal vectors to *0*- and to *2*-cells.

Regularized operations Also, it is well-known and easy to see (e.g. [23]) that a Boolean operation between regular polyhedra may give a non regular result, i.e. a point set with subsets of different dimensions. For this reason the concept of *regularized Boolean* was introduced. If $op \in \{\cup, \cap, -\}$, then the *regularized* op^* is defined as:

$$op^* := \text{clos} \circ \text{int} \circ op$$

so that, for every solid pair of the same dimension ***P*₁** and ***P*₂**, it is

$$P_1\, op^*\, P_2 := \text{clos}\,(\text{int}\,(P_1\, op\, P_2))\,.$$

Where clos and int respectively denote topological *closure* and *interior*. Therefore, in the following of this paper we only discuss algorithms for *regularized union* operation.

2.2 PRISM A-PATCHES

Prism A-patches are low-degree finite elements of algebraic surfaces, with dual implicit and rational parametric representations. In [4], [25] in input is given a matched triangulation pair $T=(T_0, T_1)$ (also called a *fat triangulation*) with attached normals at each vertex, which offers a linearization of the inner and outer boundary surfaces of a shell domain. The goal is to reconstruct a smooth fat surface whose bounding surfaces provide approximations of T_0 and T_1, respectively. Additionally mid-surfaces between the boundary surfaces may be also generated.
The matched pair of surface triangulations with normals could be obtained via several methods, including close iso-contours of volume data, point clouds, single surfaces, etc. Actually, a single triangulation *T* is sufficient, together with external normals $\mathbf{n}$,$\mathbf{n}_{ij}$,$\mathbf{n}_{ijk}$ attached to either the 0-cells (vertex normals), or to the 1-cells (edge normals, to denote the parallel transport of sharp features), or to the 2-cells (face normals) of *T*.

2.2.1 Triangular Prism A-patch

The triangular prism A-patch element is de□ned within a prism scaffold. For each triangle [$\mathbf{v}_i$,$\mathbf{v}_j$,$\mathbf{v}_k$] of a triangulation *T* of the surface, let

$$\mathbf{v}_\ell(\lambda) = \mathbf{v}_\ell + \lambda \mathbf{n}_\ell, \qquad \ell = i, j, k, \qquad \lambda \in I := [-1,1],$$

where the unit normals $\boldsymbol{n}_l$ point outward. Then define the prism

$$D_{ijk} := \{\ \mathbf{p} : \mathbf{p} = \alpha_1 \mathbf{v}_i(\lambda) + \alpha_2\ \mathbf{v}_j(\lambda) + \alpha_3 \mathbf{v}_k(\lambda), \lambda \in I\}\,, \tag{0.1}$$

where $(\alpha_1, \alpha_2, \alpha_3)$ are the barycentric coordinates of points in [$\mathbf{v}_i$,$\mathbf{v}_j$,$\mathbf{v}_k$]. The coordinate transformation from local (prism's) coordinates to global (world's) reference system can be written, by substituting $\alpha_3 = 1 - \alpha_1 - \alpha_2$, as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = D(\lambda) \begin{pmatrix} 1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \left(\mathbf{v}_i(\lambda), \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda), \mathbf{v}_k(\lambda) - \mathbf{v}_i(\lambda)\right) \begin{pmatrix} 1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}. \tag{0.2}$$

The mapping (0.2) is three-linear in $\lambda, \alpha_1, \alpha_2$.

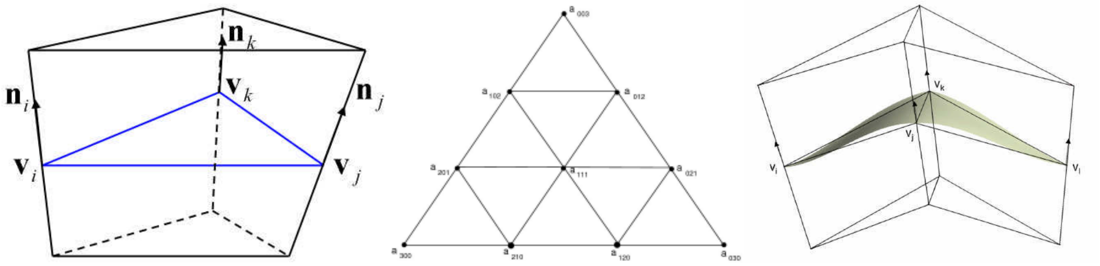

Fig. 1: (a) A prism D_{ijk} constructed based on triangle [$\mathbf{v}_i$,$\mathbf{v}_j$,$\mathbf{v}_k$]. (b) The control coefficients of a cubic Berstein-Bézier basis of function *F*. (c) Smooth connection of two prism A-patches.

The surface *S* of a patch is defined as the zero contour of a scalar field defined as a polynomial in the Benstein-Bézier (BB) form over the prism D_{ijk}:

$$F(\alpha,\lambda) = \sum_{i+j+k=n} a_{ijk}(\lambda) B^n_{ijk}(\alpha)\,, \tag{0.3}$$

where $B_{ijk}{}^n(\alpha)$ is the Bézier basis

$$B^n_{ijk}(\alpha) = \frac{n!}{i!j!k!}\alpha_1^i \alpha_2^j \alpha_3^k\,.$$

Since S passes through the vertices $\mathbf{v}_i$, $\mathbf{v}_j$, $\mathbf{v}_k$, we know that

$$a_{300} = a_{030} = a_{003} = \lambda\,. \tag{0.4}$$

To obtain C^1 continuity at the vertices, we let $a_{210} - a_{300} = \frac{1}{3}\nabla F(v_i)\cdot(\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda))$ where $\nabla F(v_i) = \mathbf{n}_i$. Therefore

$$a_{210} = \lambda + \frac{1}{3}\mathbf{n}_i\cdot(\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)) \tag{0.5}$$

$a_{120}, a_{201}, a_{102}, a_{021}, a_{012}$ are defined similarly.

The final remaining coefficient a_{111} is computed using a side-vertex interpolation method, and is given by

$$a_{111} = w_1 a^{(1)}_{111} + w_2 a^{(2)}_{111} + w_3 a^{(3)}_{111} \tag{0.6}$$

where

$$w_i = \frac{\alpha_j^2\alpha_k^2}{\alpha_2^2\alpha_3^2 + \alpha_1^2\alpha_3^2 + \alpha_1^2\alpha_2^2}, \qquad i = 1,2,3;\ i \neq j \neq k$$

Let

$$\begin{aligned} \mathbf{d}_1(\lambda) &= \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda) = B - A \\ \mathbf{d}_2(\alpha_1,\alpha_2,\alpha_3) &= \alpha_1\mathbf{n}_i + \alpha_2\mathbf{n}_j + \alpha_3\mathbf{n}_k = C \\ \mathbf{d}_3(\alpha_1,\alpha_2,\alpha_3,\lambda) &= \mathbf{d}_1 \times \mathbf{d}_2 = B\times C + C\times A \end{aligned} \tag{0.7}$$

and

$$\mathbf{c} = C(\tfrac{1}{2},\tfrac{1}{2},0),\ \ \mathbf{d}_3(\lambda) = \mathbf{d}_3(\tfrac{1}{2},\tfrac{1}{2},0,\lambda) = B\times\mathbf{c} + \mathbf{c}\times A \tag{0.8}$$

Then as given in [25] we have

$$a^{(3)}_{111} = \frac{\mathbf{d}_3(\lambda)^\top(3a_{210}B\times\mathbf{c} + 3b_{120}\mathbf{c}\times A + A\times B)}{3\,||\,\mathbf{d}_3(\lambda)\,||^2} \tag{0.9}$$

The $a_{111}{}^{(2)}$ and $a_{111}{}^{(3)}$ are defined similarly. For the surface evaluation, given the barycentric coordinates of a point $(\alpha_1,\alpha_2,\alpha_3)$ in the triangle [$\mathbf{v}_i$,$\mathbf{v}_j$,$\mathbf{v}_k$], we solve the equation $F = 0$, for λ by Newton's method, where F is defined in (0.3). Then the corresponding point is

$$(x,y,z)^\top = \alpha_1\mathbf{v}_i(\lambda) + \alpha_2\mathbf{v}_j(\lambda) + \alpha_3\mathbf{v}_k(\lambda)\,. \tag{0.10}$$

2.3 Scaffold Data Structure

To represent in memory the polyhedral envelope that contains the A-patches which cover the boundary of a curved solid, we use the WT (winged-triangle) [21] representation scheme, i.e. a boundary representation based on vertices and triangles (***0***- and ***2***-simplices) of a simplicial complex that triangulates the object boundary. This scheme provides a relational representation with tuples of constant length, that allows multishell polyhedra to be dealt with, regardless of the topological genus of their boundary surfaces. Nonmanifold objects are represented by duplicating some adjacency information. The space of mathematical models represented by the WT scheme is quite extensive. It coincides with the set of regular ***3***-polyhedra which are possibly: unconnected; unbounded (but with bounded boundary); non-manifold; multishell; with multiply connected faces, i.e. with multiple edge loops.

In particular, the WT representation can be implemented as a table in first normal form, where each tuple, indexed by the ***t***$_j$ triangle, contains:

- 3 references to incident vertices;
- 3 references to adjacent triangles.

This scheme is characterized by the design choice of making no use of Euler operators. This choice is allowed by the extreme simplicity of the data structure representing the triangulation of the boundary, and by the straightforwardness of the Boolean algorithms, discussed in the following section, where the consistency of the boundary simplicial complex is easily maintained.

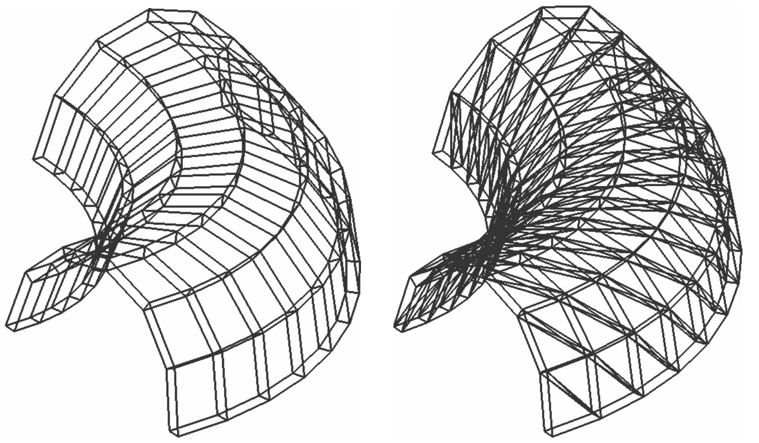
Fig. 2: (a) The scaffold of a surface; (b) the triangulated scaffold, showing the trilinear nature of each prism.

3. ALGORITHM FOR BOOLEAN SET OPERATION

Several approaches to the computation of Boolean set operators using a boundary representation can be found in the solid modeling literature (see, e.g. [3], [12], [16], [17], [18]). The very simple Boolean algorithms defined with the winged-triangle (WT) boundary representation, that does not require the use of Euler operators [8], can be found in [21].

Outline of the algorithm The main assumption we make is that the boundary of the solid is triangulated by curved algebraic patches, where each patch is single-sheeted and contained within the prismatic scaffold defined by one or two (adjacent) triangles of the linear approximation of the solid boundary. This assumption reduces the Boolean problem with curved solids to the much easier and well-known problem with linear polyhedra.

The algorithm for the regularized set union of two regular polyhedra P_1, and P_2 is composed of two main steps (see [21]). In the first step, each shell of P_1 is intersected with every shell of P_2, where a boundary shell is a maximal connected subset of the boundary. The result of this computation is the space curve where the boundary surfaces of the input polyhedra intersect. In general, this space curve is nonplanar and may be disconnected, i.e. it may contain more than one cycle (connected component). The second step of the union algorithm reconstructs the output boundary $\partial(P_1 \cup P_2)$ by subdividing the input shells in alternate patches along the intersection curve previously determined, and by choosing which subpatches must be collected into the result.

The intersection curve subdivides each intersecting shell into no more than three disjoint open surfaces: the surface *outside*, *inside* and *on the boundary* of the other polyhedron. The part of the boundary of the resulting polyhedron, generated by the intersecting shells, is very easy to set up: the corresponding algorithm is detailed in the paragraph on *partitionable shells*. The paragraph on *isolated shells* analyses how to evaluate the part of the boundary of the union where input shells do not intersect. Such isolated shells may or may not appear in the resulting boundary: to discard or to collect them in the Boolean result will depend on their position with respect to the polyhedron to which they do not belong.

Shell extraction The Boolean operations are closed in the set of polyhedra only if may regard a polyhedron as unconnected. Even a connected polyhedron may have an unconnected boundary. As an example consider an empty sphere, with an external and an internal boundary shell.

In order to perform a Boolean operation, say $P_1 \cup P_2$, the shells of both arguments must be preliminarly extracted from the data structure used as their computer representation. A very simple recursive algorithm working in linear time may be used with the WT representation, starting a new shell as containing a first boundary triangle, marking it, and extracting recursively the unmarked adjacent triangles. When the extraction stops, a shell (a maximal 1-connected boundary subset) has been extracted. A new shell can then be started from the first non marked triangle in the data structure, if any, and so on, until no more unmarked triangles remain. Let us denote with S_1 and S_2 the sets of shells of P_1 and P_2, respectively.

Shell intersection Each pair of shells $(s_i, s_j) \in S_1 \times S_2$ must be checked for empty intersection, using both global tests, e.g. checking for intersection of the containment boxes or spheres of s_i and s_j, and local tests when the former do not succeed. A local test will check for patch intersection, discussed in the next paragraph.
A shell of P_1 which does not intersect any shell of P_2 is referred to as an *isolated* shell. A shell of P_1 that intersects at least one shell of P_2 is referred to as a *partitionable* shell. Let I and T be two sets containing the isolated and partitionable shells of S, respectively, so that $S = I \cup T$. A shell of P_1 moves from the set I_1 (initially set to S_1) to the set T_1 (initially set to 0) when one of its patches gives a *valid* intersection with some patch of P_2.
An intersection between two curved triangles (i.e. two A-patches) is said *valid* when there exists points of P_1 that belong to both the *below* and the *above* subspaces of P_2, and viceversa, i.e. there exists points of P_2 that belong to both the *below* and the *above* subspaces of P_1. In other words, an intersection between two patches is valid when they really cross each other, and do not simply touch (i.e. intersect) on a subset of points (topologically: a cell) of whatever dimensionality.

Patch intersection and splitting When a valid intersection between two A-patches occurs, the intersection curve is inserted in both input A-patches, by splitting the support triangles t_1 and t_2 (and possibly their adjacent triangles) so as to split the patches into subpatches supported by two new subsets of triangles, pairwise containing a segment of the intersection curve as the common edge. The algebraic formulation of the intersection problem is discussed in Section 4.1; the numerical tracing of the intersection curve is shown in Section 4.2 , and the final triangle splitting is discussed in Section 4.3 .

Partitionable shells The intersection curves subdivide the partitionable shells into no more than three sets, which contain the surface patches that are internal, external and on the boundary of the other polyhedron, respectively. Such surfaces must be alternatively chosen to collect the whole boundary of the resulting polyhedron. In the general case, the envelope of the polyhedron $P_1 \cup P_2$ will be composed by:

(i) all the boundary patches of P_1 external to P_2 (and vice versa);
(ii) *some* of the patches laying on the boundary of both input polyhedra. In particular, the only common patches with same orientation of the external normals;
(iii) *some* of the nonintersecting input shells, that we named isolated patches (see the next paragraph).

Isolated shells An isolated shell may appear or may not appear in the resulting solid $P_1 \cup P_2$: it will be collected in the final result only if it resides in the exterior of the object to which it does not belong. More precisely: given a shell $s \in I_1$ (respectively I_2), where I_1 is the set of isolated shells of P_1 (respectively P_2), then $s \subseteq \partial(P_1 \cup P_2)$ if and only if it is external with respect to P_2 (respectively P_1). The problem of classifying a shell versus a nonintersecting polyhedron is reduced to the classi�cation of a single shell point. To solve efficiently this classification problem with respect to every cardinality of the set of patches, some sort of spatial index (say, a BSP-tree) would be very useful.

The algorithm for set union in pseudo code is thus:

```
1. Extract the shell sets $S_1$ and $S_2$ of polyhedra $P_1$ and $P_2$;
2. For each shell pair $(s_i, s_j) \in S_1 \times S_2$:
    a. If $(s_i, s_j)$ intersect then partition $s_i$ and $s_j$ into ($s_i^+$, $s_i^-$), ($s_j^-$, $s_j^+$) and $s_{ij}^0$;
       Else store $s_i$ and $s_j$ into $I_1$ , $I_2$.
3. Assemble the result $R$ (i.e. glue $s_i^+$ and $s_j^+$ along $s_{ij}^0$);
4. Add each external isolated shell of $I_1$ and of $I_2$)to the result R.
```

Operationally, step 2.a may be decomposed into:

```
2.a. If the bounding boxes or spheres of $(s_i, s_j) \in S_1 \times S_2$ intersect:
     i.   For each patch pair $(t_k, t_l) \in s_i \times s_j$, if intersect:
            i.   For all the intersection curves compute an A-spline approximation $t_{kl}^{\ 0}$;
            ii.  Partition $t_k$ and $t_l$ into A-patch sets $t_k^{\ +}$, $t_k^{\ -}$ and $t_l^{\ -}$, $t_l^{\ +}$, respectively;
     ii.  If no intersecting patch pairs are found, then store $s_i$ and $s_j$ into $I_1$ , $I_2$.
          Using the adjacency information:
            i.   Link all $t_{kl}^{\ 0}$ into loops;
            ii.  Propagate the partitioning and classification to all non intersecting
                 patches $t_k$ of $s_i$ (and $t_l$ of $s_j$);
            iii. Create the new adjacency information.
```

4. INTERSECTION METHODS

Given two solids, represented by a boundary representation of prismatic A-patches, the Boolean algorithm in section 3 has the two critical steps: (1) computation of the intersection of two patch elements; and; (2) split along the intersection into a new patch sets. In this section we describe an approximate solution. The prismatic A-patch intersection may be further decomposed into:

- algebraic formalization of the intersection problem;
- topologically sound piecewise-linear tracing of the intersection curve between two patches;
- construction and classification of new prism (sub)patches along the approximate intersection.

The Boolean algorithm will then select the appropriate (sub)patches according the classification given to compute a Boolean operation.

4.1 Intersection of A-patches

Consider two prismatic A-patches *A* and *B*. Patch *A* (respectively *B*) is built on prisms *D* (respectively *E*) of vertexes v_i, v_j, v_k (resp. w_i, w_j, w_k) and normals n_i, n_j, n_k (resp. o_i, o_j, o_k). Call the prism's local coordinates, $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ (resp. $\beta = (\beta_1, \beta_2, \beta_3)$) for the barycentric part, and, λ (resp. μ) for the height along the interpolated normal $\mathbf{n}(\alpha) = \alpha_1 \mathbf{n}_i + \alpha_2 \mathbf{n}_j + \alpha_3 \mathbf{n}_k$ (resp. $o(\alpha) = \ldots$). The global coordinates generic point in the prism is $\mathbf{v} = \mathbf{D}(\lambda)\alpha$ ($\mathbf{w} = \mathbf{E}(\mu)\beta$). The surface patch is defined as the zero-contour of scalar field $F(\alpha, \lambda) = 0$ ($G(\beta, \mu) = 0$).The scalar field may be written in B.B. form as

$$F(\alpha, \lambda) = \sum_{i+j+k=n} a_{ijk}(\alpha, \lambda) B^n_{ijk}(\alpha) \quad \left(resp. G(\beta, \mu) = \sum_{i+j+k=n} b_{ijk}(\beta, \mu) B^n_{ijk}(\beta) \right).$$

Given a certain point $\alpha(\beta)$ in the barycentric coordinate, by root finding and selecting the $\lambda \in I_\lambda$, one has a procedural parametric formulation of the surface

$$\lambda = \lambda_F(\alpha) \quad (resp. \mu = \mu_G(\beta)).$$

Restricting the attention to cubic A-patches, the B.B. coefficients are uniquely determined by the prism data (see section 2.2). With the sole exception of a_{111} (b_{111}), the coefficients are simple linear expressions $a_{ijk}(\lambda) = a^{(1)}_{ijk}\lambda + a^{(0)}_{ijk}$ .The boundary curves of a patch have a simple rational parametric form (ie. $\alpha_2 = 0 \quad \text{and} \quad \alpha_3 = 1 - \alpha_1$) :

$$\lambda = \lambda_F(\alpha_1) = \frac{\sum_{i+j=n} a^{(0)}_{ijk} B^n_{ij}(1-\alpha_1, \alpha_1)}{\sum_{i+j=n} a^{(1)}_{ijk} B^n_{ij}(1-\alpha_1, \alpha_1)} \quad \left(resp. \mu = \mu_G(\beta_1) = \frac{\sum_{i+j=n} b^{(0)}_{ijk} B^n_{ij}(1-\beta_1, \beta_1)}{\sum_{i+j=n} b^{(1)}_{ijk} B^n_{ij}(1-\beta_1, \beta_1)} \right)$$

The intersection between the two surfaces of the patches is implicitly defined by the two surface equations and by equating the coordinate transformation of the two prisms. The intersection is a system of 5 non-linear equations in 6 variables thus describing a curve.

$$\begin{cases} \mathbf{D}(\lambda)\begin{pmatrix}1\\ \alpha_1\\ \alpha_2\end{pmatrix} & = & \mathbf{E}(\mu)\begin{pmatrix}1\\ \beta_1\\ \beta_2\end{pmatrix} \\ F(\alpha,\lambda) & = & 0 \\ G(\beta,\mu) & = & 0 \end{cases}$$

The objective is to reduce the system in a minimal number of equations involving only the barycentric coordinates of one of the two prisms (i.e. ***E***).

Consider the coordinate transformation of prism $\mathbf{D}(\lambda)$ as a λ -family of linear transformations and invert keeping λ symbolic

$$\mathbf{D}(\lambda)^{-1} = \frac{1}{|\mathbf{D}(\lambda)|}\begin{pmatrix} (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)) \times (\mathbf{v}_k(\lambda) - \mathbf{v}_i(\lambda)) \\ (\mathbf{v}_k(\lambda) - \mathbf{v}_i(\lambda)) \times \mathbf{v}_i(\lambda) \\ \mathbf{v}_i(\lambda) \times (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)) \end{pmatrix}.$$

One may then compose with $\mathbf{E}(\mu)$

$$\begin{pmatrix}1\\ \alpha_1\\ \alpha_2\end{pmatrix} = \mathbf{D}(\lambda)^{-1}\mathbf{E}(\mu)\begin{pmatrix}1\\ \beta_1\\ \beta_2\end{pmatrix}$$

By bringing $|\mathbf{D}(\lambda)|$ to the left in the first row one has an equation relating in β,λ,μ **.** The second and third row define α as rationals of β,λ,μ . Denote these as $Q(\beta,\lambda,\mu)=0$ and $\alpha(\beta,\lambda,\mu)$ respectively.

An operational example is, given any point $(\overline{\beta},\overline{\mu})$ in **E** coordinates: (1) solve $Q(\overline{\beta},\lambda,\overline{\mu})=0$ for $\overline{\lambda}\in I_\lambda$; (2) compute $\overline{\alpha}=\alpha(\overline{\beta},\overline{\lambda},\overline{\mu})$.

One may use the prism coordinates equations symbolically, substituting $\alpha(\beta,\lambda,\mu)$ in $F(\alpha,\lambda)=0$, obtaining

$$\overline{F}(\beta,\lambda,\mu) = F(\alpha(\beta,\lambda,\mu),\lambda) = 0,$$

then use $Q(\beta,\lambda,\mu)=0$ to eliminate λ by resultant

$$\overline{\overline{F}}(\beta,\mu) = Res_{\lambda,0}(\overline{F}(\beta,\lambda,\mu),Q(\beta,\lambda,\mu)) = 0.$$

The intersection equations are now reduced to two equations in the local coordinates of prism ***E***

$$\begin{cases} \overline{\overline{F}}(\beta,\mu) & = & 0 \\ G(\beta,\mu) & = & 0 \end{cases}$$

Conceptually, one may proceed further, and use the parametric form of the surface $\mu=\mu_G(\beta)$ and reduce the intersection to a curve in the barycentric coordinates β of prism **E**.

$$F_G(\beta) = \overline{\overline{F}}(\beta,\mu_G(\beta)).$$

Given any line in the barycentric coordinates β (eg: a linear constraint) one may by root finding compute the intersection point. However, due to the procedural nature of $\mu_G(\beta)$ one may only proceed numerically. The intersection between the surface of patch **A** and a border (ie. $\beta_2=0$) of **B** may be fully developed symbolically by plugging the rational $\mu(\beta_1)$ into $\overline{\overline{F}}(\beta_1,\mu)$ obtaining $F_G(\beta_1)=\overline{\overline{F}}(\beta_1,\mu_G(\beta_1))$.

Symmetrically the same intersection curve may be in defined in domain α of prism **D**.

4.2 Intersection Approximation

The intersection points between the boundary of one patch and the surface of the other may be formulated as the roots of univariate polynomials, albeit of high degree:

$$F_G(\beta_1) = 0, \quad F_G(\beta_2) = 0, \quad F_G(\beta_3) = 0, \quad G_F(\alpha_1) = 0, \quad G_F(\alpha_2) = 0, \quad G_F(\alpha_3) = 0.$$

Methods to approximate *all* the roots of a polynomial exist. This is a powerful starting point for a topologically accurate tracing of the intersection curve. Even with the aid of computer algebra systems, the symbolic development of such polynomials for a pair of generic patches, has been found too cumbersome. On the other hand the computation of such polynomials for each specific pair of patches is too time consuming at implementation level.

In a prototype implementation a purely numerical approach has been used. Given a certain point $\overline{\beta}$, in **E** barycentric coordinates:

(i) compute the height $\overline{\mu}$ along the interpolated normal of the surface *G* using the procedural form $\overline{\mu} = \mu_G(\overline{\beta})$;

(ii) solve $Q(\overline{\beta},\lambda,\overline{\mu}) = 0$ for λ by root finding and selecting $\overline{\lambda} \in I_\lambda$;

(iii) compute the corresponding barycentric coordinates in prism **D** by $\overline{\alpha} = \alpha(\overline{\beta},\overline{\lambda},\overline{\mu})$;

(iv) Compute $F(\overline{\alpha},\overline{\lambda})$.

This procedure effectively computes $F_G(\overline{\beta})$. Furthermore, the membership of the point in the interior of prism **D** is determined. The computation may be used iteratively with a numeric root finding algorithm (such as the false position method) to find an intersection point ($F_G(\beta) = 0$) along any line segment in the β barycentric domain.

Operationally, the barycentric domain of *both* prisms is subdivided. The particular subdivision is not important, for simplicity the standard decomposition along iso-coordinates is used (Fig. 3a).

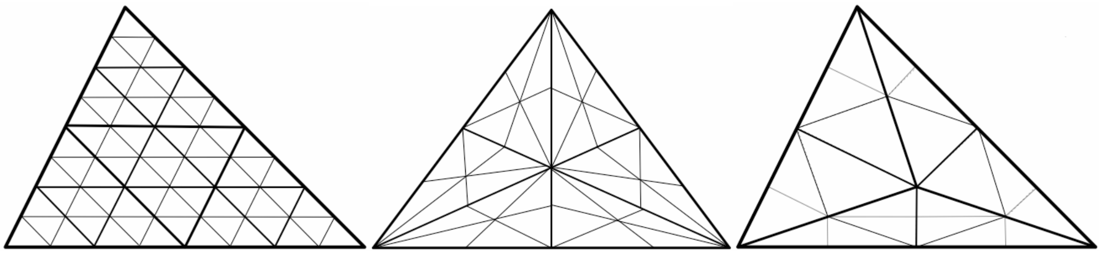

Fig. 3: Subdivision along: (a) iso-coordinates; (b) barycenter; (c) orthocenter.

Without loss of generality consider the prism **E**, the same operations are symmetrically done in the other prism **D**. For each vertex of the subdivision the corresponding value of the surface scalar function $F_G(\beta)$ is computed. Each sub-triangle may either have no sign change (even number of intersections) or a sign change on two sides (odd number of intersections). A simplistic approach is used and only one intersection in case of a sign change is searched. For each intersection point save the barycentric coordinates β , the height μ, and the corresponding coordinates α and λ in the other prism.

All intersection points in the same sub-triangle will be considered in the same connected component of the intersection, thus a tracing of the intersection curve is possible. For the correctness of next steps it is important that: (1) the trace of the intersection is done on both prisms; and; (2) both tracings contain the *same* points. (Fig. 4 and 5)

An advantage of this simplified algorithm is that the topology of the intersection in each sub-triangle is the same as the intersection between two triangles and apt to be processed by the Boolean algorithm described in section 3 . Topological errors are possible but their size is limited by subdivision granularity. (Fig. 6)

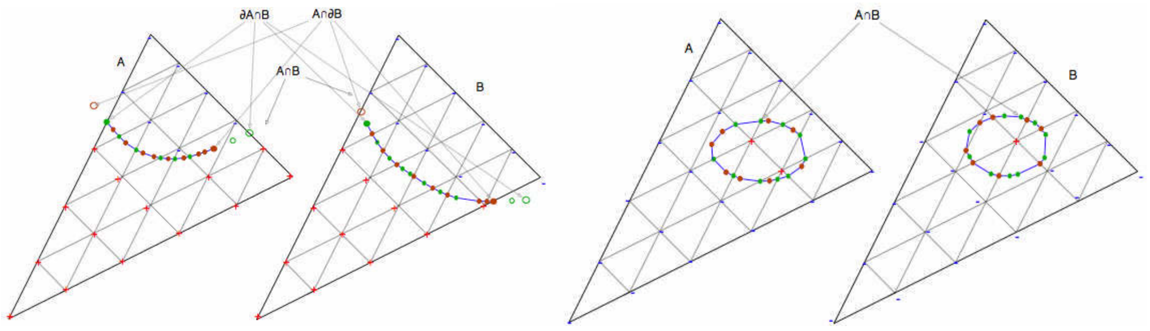

Fig. 4: Tracing the intersection on both domains.

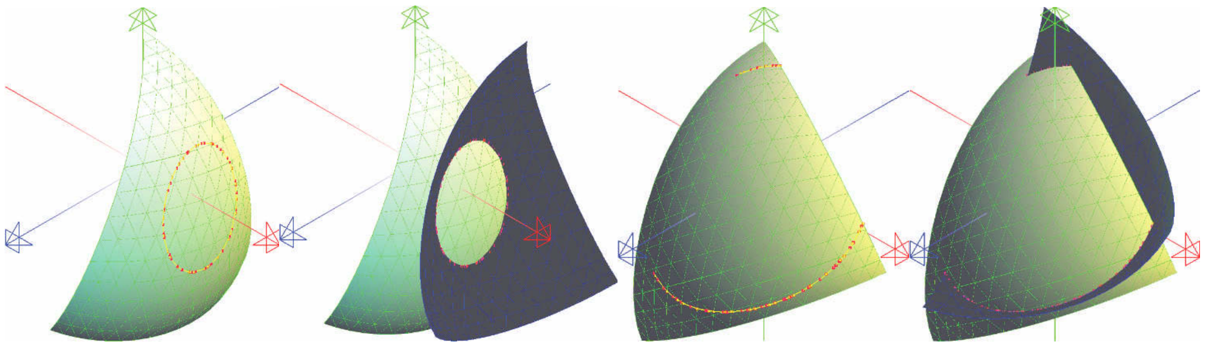

Fig. 5: Tracing the intersection points on both domains: (a,b) a closed intersection curve between two A-patches; (c,d) two unconnected segments of the intersection curve.

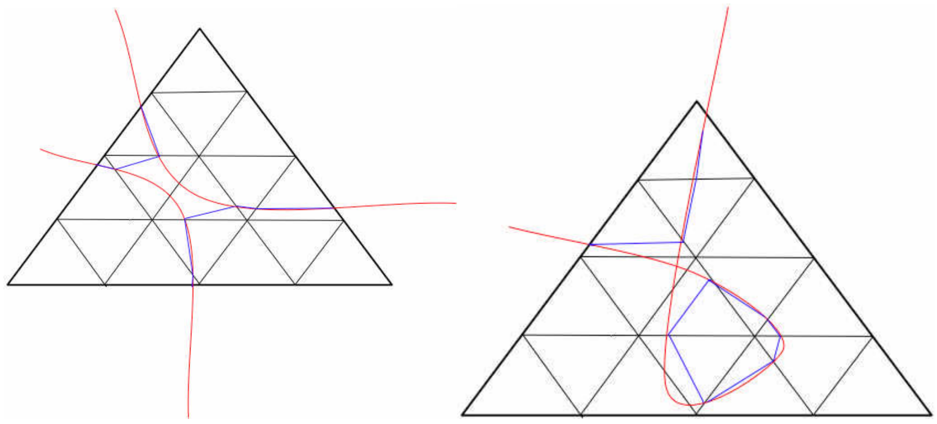

Fig. 6: Topological approximation.

4.3 Prism Scaffold Reconstruction

The final step of our intersection algorithm is to reconstruct triangular (and hence scaffold) support, conforming to the topology of the intersection curve, for all the prism A-patches, that constitute the result of the Boolean operation. Since the topology of the intersection curve is either an open or a closed curve in both the patch domains, the necessary conforming triangulation can easily be affected in either or both domains.

In both the top and bottom rows of Figure 7 the leftmost figures show the intersection topology of the two A-patches (shaded differently). The middle figures (top and bottom) depict the result of the union operation. The rightmost figures show the patch decomposition into triangles and quads. The quads are further subdivided into two triangles,

using any diagonal. The prism scaffolds are easily erected on each triangle based on the normals defined at the vertices.
Note, for vertices lying on the intersection curve, there exist two vertex normals (one from each initial A-patch). Hence these vertices are part of two prism scaffold edges.
Note that two adjacent prism scaffolds may intersect each other, since the resulting patches are no longer C^1.

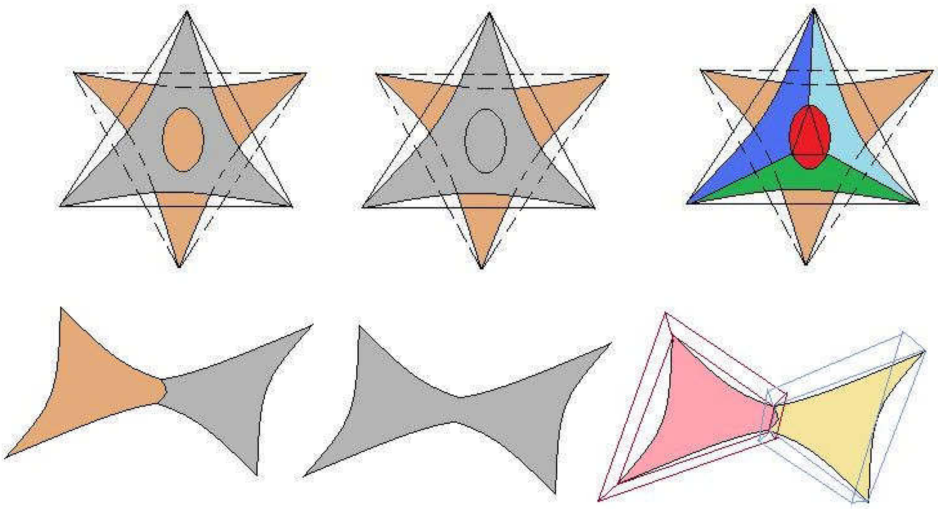

Fig. 7: Examples of conforming triangular supports of the prism A-patches that constitute the union of original twin A-patches. Shown are parts of the scaffolds of the A-patches for two different topologies of the intersection curve. The triangulation of the quad patches are not shown, just to maintain picture clarity.

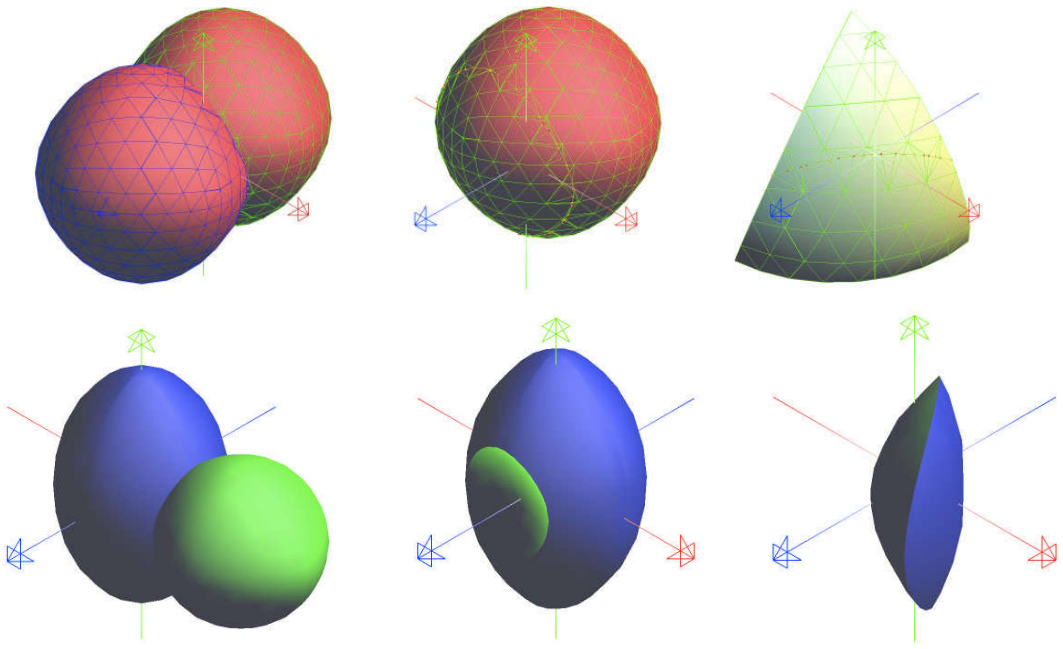

Fig. 8: [Top row, Left]: Union of two spheres; [Top row, Middle]: The intersection curve shown on one of the spheres; [Top row Right]: Zoom of the intersection curve shown on a single prism A-patch of the Sphere;[Bottom row, Left]: Union of a green sphere with a blue free-form solid ; [Bottom row, Middle]: The Difference of the green sphere from the Blue solid; [Bottom row Right]: Intersection solid region

5. APPLICATIONS AND EXAMPLES

Molecular Models Physical models of molecular models via 3D printing have seen increasing use in bio-chemistry kits, and more recently in tangible augmented interfaces, for experiential understanding of the complexity of molecular interfaces, especially in docked complexes. Several CAD operations need to be performed on molecular models, including Boolean set operations, thereby allowing, for example, the unioning of bonds, and differences to create holes, before the 3D auto-fabrication process [10]. In Figure 9 we show two ligands proteins and the resulting complex after a union, intersection and difference operation. Our prototype implementation has been used to compute the actual intersection and union in the rest configuration. The reader should notice the small intersection curves, drawn in color (magenta) on Figures 9 b, c, e and 9f.

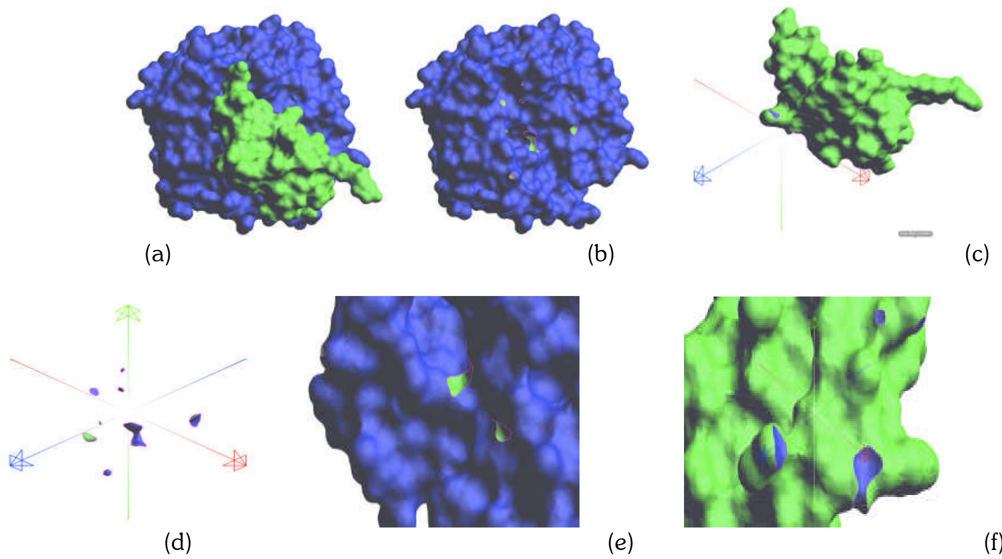

Fig. 9: The molecule M_1 with another molecule M_2: (a) the union of M_1 and M_2, (b) the difference $M_1 - M_2$, (c) the difference $M_2 - M_1$, (d) the intersection of M_1 and M_2, (e) zoom of the molecule M_1 with the intersection curves and (f) zoom of the molecule M_1 with the intersection curves.

6. CONCLUSION

In this paper we have introduced a novel approach to Boolean operations with curved solids whose boundary is triangulated by cubic A-patches, by using a boundary representation of the solids. We have also discussed a simple and robust method to trace the intersection curves between two triangular or quadrilateral prismatic A-patches, while at the same time locally refining the support triangulation. The prototype implementation provides for the computation of Boolean operations between two free-form solids, modeled by A-patches. In the near future we plan to significantly improve the efficiency of this approach with spatial progressive indices, and multiresolution evaluators, with local refinement. Another interesting research direction consists in the direct modeling within both space and time dimensions. In fact, the implicit geometric approach used here and the associated linear scaffolding can be extended quite easily to work in higher dimensions. The most flexible finite element techniques for evolution equations are based on spacetime discretizations, whereby the space and time domains are subdivided into finite elements. Evolution equations are then discretized in both space and time on each element through low-order polynomial approximation and numerical quadrature. Treating space and time in a unified way in the discretization provides a framework for both mesh refinement and coarsening where needed in both space and time, driven by rigorously-derived *a posteriori* error indicators.

7. ACKNOWLEDGEMENTS

This research was supported in part by NSF grant CNS-0540033, NIH contracts P20-RR020647, R01-EB00487, R01-GM074258, R01-GM07308 and by MIUR Finanziamento Dottorato SICS.

8. REFERENCES

[1] Bajaj, C.; Chen, J.; Xu, G.: Modeling with cubic A-patches, ACM Trans. Graph. 14(2), 1995, 103–133.

[2] Biermann, H.: Kristjansson, D.; Zorin, Z.: Approximate Boolean operations on free-form solids, SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM Press, 2001, 185-194.

[3] Braid, I.: On storing and changing shape information, SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques (New York, NY, USA), Acm, 1978, 252-256.

[4] Bajaj, C.; Xu, G.: Smooth shell construction with mixed prism fat surfaces, Geometric Modeling (London, UK), Springer-Verlag, 2001, 19–35.

[5] Bajaj, C.; Xu, G.; Holt, R.; Netravali, A.: Hierarchical multi-resolution reconstruction of shell surfaces, Computer Aided Geometric Design, 19, 2002, 89-112.

[6] Bajaj, C.; Chen, J.; Xu, G.: Modeling with C^2 Quintic A-patches, Fourth SIAM Conference on Geometric Design, Nashville, TN, 1995.

[7] Dahmen, W.: Smooth piecewise quadric surfaces, Mathematical methods in computer aided geometric design, Academic Press Professional, Inc., San Diego, CA, USA, 1989, 181-193.

[8] Eastman, C.: Weiler, K.: Geometric Modeling Using the Euler Operators, Institute of Physical Planning, Carnegie-Mellon University, 1979.

[9] Garcia, A.; De Miras, J.-R.; Feito, F.: Adaptive trimming of cubic triangular Bézier patches, SIACG '06: Ibero-American Symposium in Computer Graphics (Santiago De Compostela, Spain), 2006.

[10] Gillet, A.; Sanner, M.; Stoffler, D.; Goodsell, D.; Olson, A.: Augmented Reality with Tangible Auto-Fabricated Models for Molecular Biology Applications, Proc. of IEEE Conference on Visualization, 2004, 235-242.

[11] Guo, B.: Modeling arbitrary smooth objects with algebraic surfaces, Ph.D. thesis, Computer Science Department, Cornell University, Ithaca, NY, USA, 1992.

[12] Hoffmann, C.: Geometric and solid modeling: an introduction, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989.

[13] Keyser, J.; Culver, T.; Foskey, M.; Krishnan, S.; Manocha, D.: Esolid - a system for exact boundary evaluation, SMA '02: ACM Proceedings of the seventh ACM symposium on Solid modeling and applications (New York, NY, USA), 2002, 23–34.

[14] Krishnan, S.; Gopi, M.; Manocha, D.; M. Mine, M.: Interactive boundary computation of Boolean combinations of sculptured solids, Computer Graphics Forum, 16(3), 1997, 67-78.

[15] Linsen, L.: Netbased modelling, Proceedings of Spring Conference on Computer Graphics (SCCG) (Comenius University, Bratislava, Slowakei) (Bianca Falcidieno, ed.), 2000, 259-266.

[16] Laidlaw, D.; Trumbore, W.; Hughes, J.: Constructive solid geometry for polyhedral objects, SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM, 1986, 161–170.

[17] Mäntylä, M.: Introduction to Solid Modeling, WH Freeman & Co. New York, NY, USA, 1988.

[18] Masuda, H.: Topological operators and Boolean operations for complex-based nonmanifold geometric models, Computer Aided Design, 25(2), 1993, 119-29.

[19] Nielson, G.: The side-vertex method for interpolation in triangles, J. Approx. Theory, 25, 1979, 318-336.

[20] Paoluzzi, A.; Bernardini, F.; Cattani, C.; Ferrucci, V.: Dimension-independent modeling with simplicial complexes, ACM Transactions on Graphics, 12(1), 1993, 56-102.

[21] Paoluzzi, A.; Santarelli, A.; Ramella, M.: Boolean algebra over linear polyhedra, Computer-Aided Design 21(8), 1989, 474-484.

[22] Requicha, A.: Mathematical models of rigid solid objects, Technical Memo 28, Production Automation Project, University of Rochester, Rochester, NY, November 1977.

[23] Requicha, A.: Representations for rigid solids: Theory, methods, and systems, Computing Surveys, 12(4), 1980, 437-464.

[24] Xu, G.; Huang, H.; Bajaj, C.: C^1 modeling with A-patches from rational trivariate functions, Computer Aided Geometric Design, 18(3), 2001, 221-224.

[25] Zhao, W.; Xu, G.; Bajaj, C.: An algebraic spline model of molecular surfaces modeling, ACM Solid and Physical Modeling Symposium (Beijing, China), ACM Press, 2007, ACM Symposium on Solid and Physical Modeling 2007.