# Nonmanifold Modeling: An Approach Based on Spatial Subdivision

Paulo Roma Cavalcanti [a], Paulo Cezar Pinto Carvalho [b], Luiz Fernando Martha [c]

[a] *Institute of Mathematics, UFRJ–Universidade Federal do Rio de Janeiro, 21945-970 Rio de Janeiro, RJ, Brazil*

[b] *IMPA–Instituto de Matemática Pura e Aplicada, Estrada Dona Castorina, 110, 22460-320 Rio de Janeiro, RJ, Brazil*

[c] *Department of Civil Engineering, PUC-Rio–Pontifícia Universidade Católica do Rio de Janeiro, Rua Marquês de São Vicente, 225, 22453-900 Rio de Janeiro, RJ, Brazil*

This paper deals with the problem of creating and maintaining a spatial subdivision, defined by a set of surface patches. The main goal is to create a set of functions which provides a layer of abstraction capable of hiding the geometric and topological problems which occur when one creates and manipulates spatial subdivisions. The study of arbitrary spatial subdivisions extends and unifies the techniques used in nonmanifold solid modeling and allows the modeling of heterogeneous objects.

*Key words:* Nonmanifold Modeling; Heterogeneous Object Modeling; Spatial Subdivision.

## 1 Introduction

Researchers in Computer Graphics have been constantly looking for tools for modeling real objects. Such tools must provide at least three things:

- a representation scheme, based on a mathematical model, adequate for objects realizable in two and three dimensions;
- a data structure to store the representation of a valid object;
- and a practical manner for creating a model on a computer from scratch.

A lot of effort has been spent in finding solutions for each of these problems, and the integration of individual solutions into a single environment is the main challenge from an application point of view.

Two different strategies have been traditionally proposed to model solid objects. In the first strategy, one aims to represent a solid through an explicit description of its boundary. These are the so-called boundary representations (BRep) [1], which are based on data-structures that describe the adjacency relationships of the vertices, edges and faces of the solid. The other possibility is the CSG approach [2], which consists in representing a solid as a result of a sequence of set operations performed on simple primitive solids. These, in turn, are usually represented as the intersection of a finite set of half-spaces determined by certain surfaces. BRep and CSG have complementary advantages and disadvantages [3], and much work has been done about converting from CSG to BRep [4] and, more recently, from BRep to CSG [5–7]. Both types of conversion use the fact that the surfaces describing the faces are present in both representations, either explicitly represented (BRep) or as defining primitive solids (CSG).

Traditional BRep and CSG techniques apply when one just needs to look at an object as inducing a three-part space decomposition: its interior, its exterior and its boundary. The motivation for this work comes from the fact that, for many interesting applications, this is not enough. For instance, sometimes it is necessary to represent objects made of several materials with different properties (e.g., semiconductor circuits, motors, reinforced concrete structures, and airplanes) or objects possessing many regions (e.g., finite element meshes). In these situations, one would like to have not only a representation for each part of the model but also a description about the way these parts are connected to each other. In general, traditional modeling systems have a hard time modeling contact relationships between solids.

The basic idea in this work is to look at complex, heterogeneous objects as defining a spatial decomposition. In order to represent such objects we adopt a two-step process. The first step consists in obtaining a geometric and topological description of the spatial decomposition induced by the curves and surfaces that separate the several parts of the object and separate the object from its exterior. In the second step, we assign attributes to each topological element (vertex, edge, face, region) of the resulting decomposition. These attributes may specify, for example, if a given element belongs to the object and, if so, to which specific part it belongs. They may also contain information specific to each kind of application. For instance, in geographical maps it is necessary to attach attributes to cells (states, boundaries, rivers, etc.) of the decomposition. In engineering applications, such as systems for stress analysis, it is required to apply loads to faces or edges of a model or to attach a material property to a region.

This paper advocates the use of spatial subdivisions to model complex objects in a uniform and coherent way. The study of representation and modeling methods based on arbitrary spatial subdivisions [8–12] generalizes and unifies the so called nonmanifold modeling techniques [13,14] (i.e., the techniques used to model objects that are not necessarily two-manifolds embedded in three dimensional space).

The proposed methodology is based on the creation and maintenance of three-dimensional spatial subdivisions defined by a set of surface patches. Whenever the geometrical description of a new patch (to be inserted into the subdivision) is given, that patch is automatically subdivided into simple patches, which can be added to the data structure without violating any geometrical or topological constraint. The main goal is to provide a layer of abstraction hiding the topological and geometrical problems which occur when one creates and manipulates spatial subdivisions.

Although complex objects can be described, in principle, by its defining surfaces, in practice one needs easier ways to define such objects. This also happens in the traditional solid modeling context. Modeling systems based on BRep usually are capable to perform set operations, which enables one to use CSG-like operations to define solids, while retaining the full boundary representation. An analogous methodology is used here. We discuss how to implement a set of non-regular construction operators [15] in order to supply the user with a modeling process for complex, heterogeneous objects analogous to CSG.

The material is organized in six sections. In Section 2, we define spatial subdivisions, and discuss possible ways of representing them. In Section 3 we present the chosen representation and the operators to manipulate it. Although it is simple (and convenient) to create planar subdivisions by adding a curve segment at a time [16], the task of creating a spatial subdivision is much more complex. Thus, in Section 4, we discuss the problems arising when creating a spatial subdivision by inserting a surface patch at a time. In most cases, this is not a practical way to create spatial subdivisions from scratch. However, this procedure can be useful for converting from other types of representation. For interactive applications we discuss, in Section 5, higher level constructive geometry modeling tools for heterogeneous objects; we also comment on how to integrate those tools to existing BRep systems. Finally, in Section 6 we summarize the results and suggest directions for further research.

## 2 Representation of Spatial Subdivisions

An example of a simple spatial subdivision is shown in Fig. 1. In this subdivision, the space is divided into six bounded regions and one unbounded region. Each region is delimited by a set of shells and each shell is composed by a connected set of faces and/or wireframes.

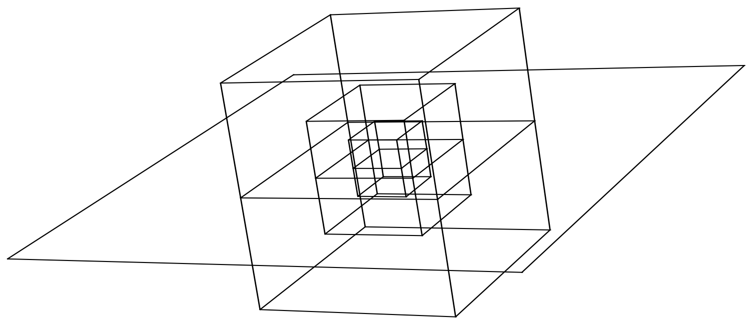
Fig. 1. A Subdivision of $\Re^3$.

A pertinent problem is: given a set of surface patches (usually given in a parametric form), how can one obtain the spatial subdivision determined by these patches? In this work, we describe a representation scheme intended to create and maintain spatial subdivisions, allowing the insertion of new patches in real time. We assume that the surface containing each patch is of bounded variation. This means that every line intersects the surface in a finite number of points and every plane in a finite number of curves [13]. This requirement is satisfied by the types of surfaces usually employed in CAGD, such as algebraic surfaces, splines or NURBS.

Although a complete geometric description carries all information about the geometric shapes of the spatial subdivision elements and their positioning in space, it is better to have both geometrical and topological information in the representation of the subdivision. For example, from the geometric description of two surface patches, their intersection curve can be found when necessary. However, this determination requires some processing, which generally is time consuming. A representation for spatial subdivisions containing explicitly all intersections not only ensures that the computation of these intersections can be done just once (when the subdivision is created), but also avoids numerical error propagation.

The basic idea is to consider the decomposition of the space in disjoint portions called *cells*, each one homogeneous in dimension and satisfying the condition that the intersection of the boundaries of any two cells is necessarily equal to the union of other cells of the decomposition. Rossignac and O'Connor [17] have proposed the concept of a *Geometric Complex* to formalize appropriately this idea.

## 3 Complete Geometric Complex

Geometrical complexes are not required to cover the entire space. For our purposes, it is convenient to restrict attention to *complete* geometrical complexes (or *CGCs* for short), in which the union of all cells is the entire space $\Re^n$. CGCs are very general and are defined in spaces of arbitrary dimension. For this reason, adjacency relationships for arbitrary CGCs are necessarily very general and cannot take advantage of the special structures present in two and three dimensions. These special structures are captured through the so-called topological representations. Since in this work we are interested only in subdivisions of three dimensional space, we shall use the term CGC to denote complete geometric complexes in three dimensions.

To represent the topology of a CGC means to represent the adjacency information of its cells (i.e., information about topological proximity and ordering of cells). The main benefit of explicitly storing the topology of a CGC in the representation is the possibility of obtaining more efficient geometrical algorithms.

We use in this paper the wide-spread BRep terminology for topological elements [1] (vertex, edge, loop, face, shell, and region). Additionally, we say that two faces $S$ and $S'$ of a CGC are *linked* if there is a sequence of faces $F_1$, $F_2$, ..., $F_n$ in the CGC such that $F_1 = S$, $F_n = S'$, and the boundaries of $F_i$ and $F_{i+1}$ have a non-empty intersection, for each $i = 1, 2, \ldots, n-1$. If each intersection has at least one edge, then $S$ and $Q$ are *strongly linked*; otherwise, they are *weakly linked.*

Spatial subdivisions commonly contain edges having more than two incident faces, or volumes connected by a single vertex. Any representation for CGCs must somehow be capable of handling these and other nonmanifold conditions.

Several works have presented methods to represent spatial subdivisions. Rossignac and O'Connor [17] have treated the general problem of representing $n$-dimensional objects, possibly with internal structures. Some data structures used in nonmanifold solid modeling [18,9–11] represent, in a general way, the adjacency relationships of three dimensional objects not necessarily homogeneous in dimension. Here, we use the Radial Edge (RED) data structure proposed by Weiler [18].

RED explicitly stores the two uses (sides) of a face by the two regions (not necessarily distinct) that share that face. Each face use is bounded by one or more loop uses, which in turn are composed by an alternating sequence of edge uses and vertex uses (Fig. 2). Vertex uses are necessary to store nonmanifold conditions at vertices.

The loop uses in a face use must be coherently oriented in the data structure to guarantee global consistency. We choose to impose a clockwise orientation to the outer loop of a face use (when observing that loop from the region that uses it, see Fig. 2). Internal loops are oriented counter-clockwise. With this convention, the "signed volume" of each bounded region is positive [1]. This

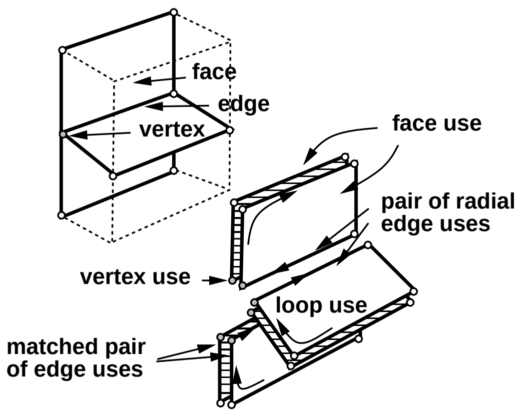

Fig. 2. Use of topological elements in RED.

fact can be used to ensure data structure consistency when including a new face if that originates a new region.

Topological data structures are too complex to be manipulated directly. Weiler has introduced a set of operators [19] that provide a high level method to access RED[1]. These operators are divided in two groups. The first group has operators that act on faces of a CGC and are analogous to Euler operators [1]. The second group has operators that are capable of creating wireframes and adding faces, which are "stitched" to specified edges or wireframes.

The enclosing of a new region is accomplished by Weiler operator `make_face`, which employs a traversal algorithm. Using edge adjacencies, the face uses that can be reached from one side of the new face are traversed and marked. If the other use is not marked at the end of the traversing, a new region has been created.

A similar process is used to distribute the face uses between the new region and the old region. If the faces delimiting these regions are not strongly linked (the connection is just by vertices or wireframes), some face uses are not traversed and do not belong to any region (Fig. 3). For this reason, a new output parameter was added to the original operator specified by Weiler. This new parameter is a list (possibly empty) with the unclassified face uses. This classification cannot be done using only topological information: geometrical tests are required to decide the region that contains each of those faces.

In our implementation, operator `make_face` was extended in relation to Weiler's to support faces with disconnected boundaries (multi-loops). This operator deletes the shells which are joined in consequence of the addition of the new face. We allow loops with dangling edges.

[1] Although some work has been done in presenting nonmanifold operators in a more rigorous form [20], there is not yet a definitive work about the subject.

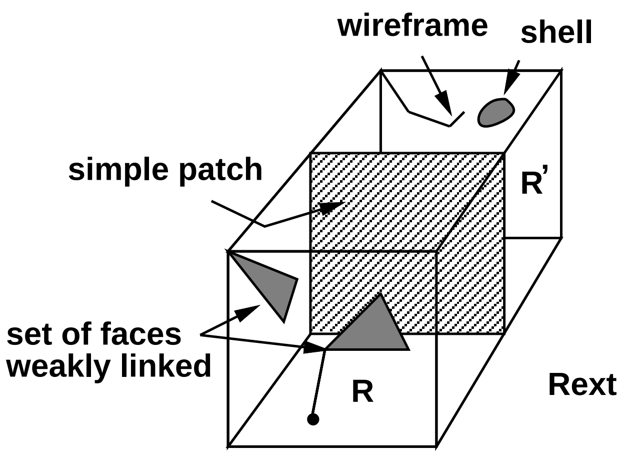

Fig. 3. Creation of a new region.

## 4 Incremental CGC Creation

In this section, we consider the incremental creation of a CGC. That is, we consider the inclusion of a patch at a time, with the corresponding updating of the data structure after each insertion. The insertion of a new patch is a fundamental operation executed by the proposed spatial subdivision scheme.

We assume that a surface patch is orientable, without singularities, connected and with boundary. A procedure to insert patches into a CGC must guarantee its topological and geometrical consistency after the insertion. As in two dimensions, where topological consistency is guaranteed by the use of Euler operators [1], topological consistency of a spatial subdivision is naturally maintained by the appropriate use of Weiler operators. To achieve this, an incoming patch must be subdivided into a set of patches, called *simple patches*, that are completely contained in regions of the CGC.

To subdivide a patch $S$ it is necessary to find the faces $f_i$ of the CGC crossed by $S$ and the curve segments determined in each intersection (Fig. 4). These segments are used to refine $S$ and each $f_i$. This refinement can be done by inserting each curve segment into the appropriate faces. Since the geometrical support of a face is homeomorphic to $\Re^2$, the method described in [16], to include a simple segment in a (planar) face, can be readily adapted to deal with this case.

Once both the incoming patch and the faces crossed by it have been subdivided, we have a set of simple patches that fit in the CGC. Thus, the problem is now reduced to inserting a new simple patch $S$. It is necessary to find:

– the vertices $v_i$ and edges $e_i$ of the CGC that must be stitched in the bound-

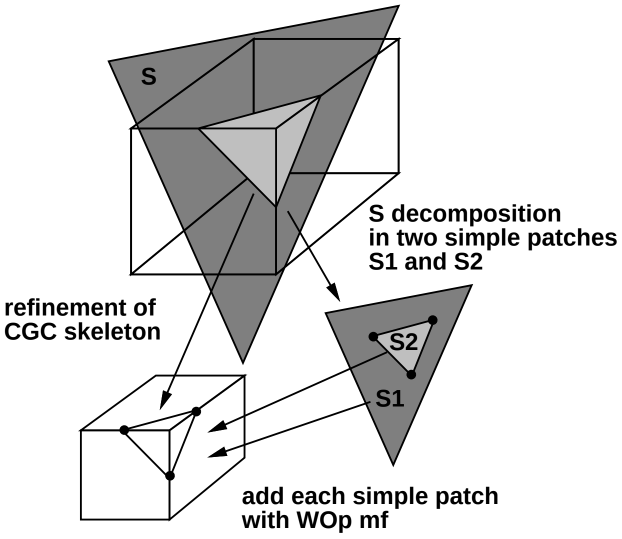

Fig. 4. Insertion of a patch in a CGC.

ary of $S$.
– for each $e_i$, the face that succeeds $S$ in the ordered cycle of faces about $e_i$.
– the region $R$ containing $S$.

The first step is to find the vertices of the CGC geometrically coincident with vertices of $S$. These vertices are stored in a list $V_l$. The second step is to find the edges of CGC geometrically coincident with edges of $S$. Only edges linking vertices in consecutive positions of $V_l$ need to be considered. These edges are kept in a list $E_l$.

If, at the end of this process, $E_l$ is empty, then we conclude that $S$ was either disconnected from the CGC or weakly connected to it (linked only by vertices or wireframes). In both cases, $R$ is the region containing an arbitrary point of an edge of $S$. If $E_l$ is not empty then, for each edge $e_i$ in $E_l$, the face succeeding $S$ in the ordered cycle of faces about $e_i$ must be found. When the faces are planar, it is enough to consider, in each face $f$ incident to $e_i$, a vector $V(f)$ which is perpendicular to $e_i$. The face $f'$ succeeding $S$ is the face for which $V(f')$ determines a minimum oriented angle with $V(S)$. This face is stored in a list $F_l$. The face $f''$ that precedes $S$ is the face for which $V(f'')$ determines a maximum oriented angle with $V(S)$. $R$ is the region delimited by each face succeeding and preceding $S$ in the ordered cycle of faces about each $e_i$ (Fig. 5).

The above method can be adapted for the case where the faces are not planar, by taking $V(f)$ contained in the plane tangent to $f$ at a point of $e_i$ and perpendicular to the vector tangent to $e_i$ at the same point.

The next step for the insertion of $S$ is to add to the CGC each $v_i$ and each $e_i$ that does not belong to the CGC using the appropriate Weiler operator

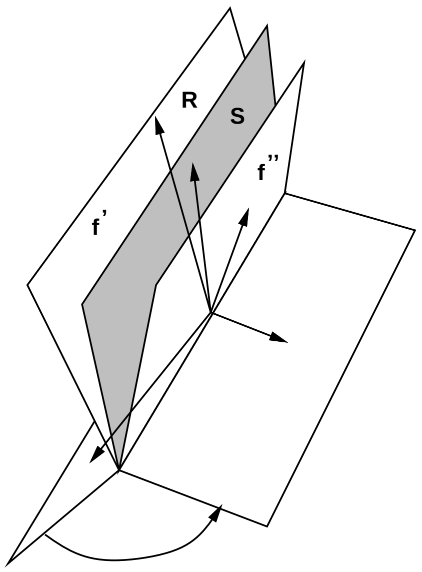

Fig. 5. Ordering of faces about edges.

(`make_edge` or `make_edge_and_vertex`), which creates wires corresponding to the new edges. Finally, the face corresponding to the simple patch is inserted with the operator `make_face`.

### *4.1 Creation of a New Region*

The addition of a simple patch $S$ may produce a new region $R'$ in the spatial subdivision. This occurs when the inclusion of $S$ causes the subdivision of $R$ and of one of its shells (which will be called the *construction shell*). It is necessary to distribute the several shells of $R$ between $R$ and $R'$, in order to keep geometrical and topological consistency.

First, it is necessary to determine which of the two portions of the subdivided shell delimits the new region $R'$. We choose one shell arbitrarily and calculate the "signed volume" of its region. If the sign is positive then the choice was correct. Otherwise, the choice was incorrect and the other portion of the construction shell becomes the new shell of $R'$.

The region $R'$ may also possess (Fig. 3):

- a set of faces weakly linked to it (meaning that possibly they are not associated with its outer shell).
- some shells or wireframes of $R$.

In each case, the following procedure must be executed to correct the representation:

- for each face of $R$ weakly linked to the construction shell, check if any point of any of its edges is in $R'$. If so, add the face to the outer shell of $R'$. Otherwise, add it to $R$.
- for each wireframe of the construction shell, check if an arbitrary point of the wireframe is in $R'$. If so, move the wireframe to $R'$.
- for each shell of $R$ (except its outer shell and the construction shell) check if an arbitrary vertex of the shell is in $R'$. If so, move the shell to $R'$.

The problem of point-in-region testing can be solved using techniques similar to those used in the two dimensional version of the problem [21]. It is possible to use an algorithm that counts the number of intersections of a ray, starting at point $p$, against the faces of region $R'$ (care should be taken in treating the difficulties introduced when the ray crosses an edge or a vertex on the boundary of $R'$). Another possibility is an algorithm based on the sum of solid angles defined by $p$ and the faces of $R'$ [22].

## *4.2 Intersection of Faces*

The most time-consuming step in the insertion of a new patch is the computation of intersections involving the patch and the existing faces of a CGC, which is done to produce a set of simple patches to be added to the CGC. The complexity of this step depends on face geometry. Although we have considered arbitrary surface geometries in the conception of the proposed architecture, our current implementation supports only planar faces and straight edges. Below, we describe an algorithm that finds the intersection of two planar faces.

Given two planar faces $A$ and $B$, possibly with multi-loops, the goal is to determine the vertices and edges that must be created (in $A$ and $B$) to make those faces compatible with each other (Fig. 6). The algorithm traverses each edge use on the boundary of each face (dangling edges are traversed twice), subdivides the intersection line $L$ of the planes containing each face, and determines several segments given by pairs of consecutive intersection points. Then each segment is classified in each face as being inside the face, outside the face, or on some edge. Based on this classification, it is possible to determine which segments correspond to new edges in each face. For instance, in order to determine an edge on $A$, a segment must be inside $A$ and, at the same time, must be inside $B$ or contained in one of their edges (this is the case of segment 2 in Fig. 6, which is an edge to be created on $A$). There are other criteria to determine which intersection points (on $L$) generate vertices on a face. A detailed description of the algorithm can be found in [23].

The algorithm above assumes that the two intersecting faces are not coplanar. When this is not the case, the intersection of the faces is no longer contained

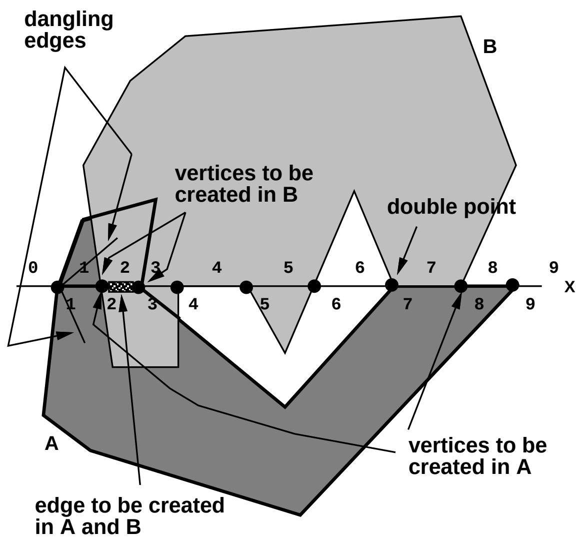
dangling edges
B
vertices to be created in B
double point
0 1 2 3 4 5 6 7 8 9
X
1 2 3 4 5 6 7 8 9
A
vertices to be created in A
edge to be created in A and B

Fig. 6. Intersection of two faces $A$ e $B$.

in a straight line. In this case, it suffices to create, in one of the faces, vertices and edges corresponding to the portions of the edges of the other face which are contained in the interior of the first face.

### *4.3 Time Estimation for the Creation of a CGC*

Consider, for the moment, that a CGC is created from simple patches that fit into it. This occurs, for instance, when the geometric description of the faces of a CGC is known and one wants to obtain its topological structure. This requires checking, for each new patch $S$, which of its vertices are already in the CGC. If $S$ has $m$ vertices and the CGC has $n$ vertices, then, in the worst case, $(m*n)$ comparisons are necessary. Identifying the edges of $S$ that coincide with edges of the CGC is simpler. The edges of $S$ join pairs of consecutive vertices of $S$, which at this point will already be in the CGC. Assuming straight edges for each one of these edges, it suffices to check whether some edge of the CGC has the same endpoints. This takes time proportional to the number of edges incident to the endpoint vertices, which typically is small. After vertex and edge identification, it is still necessary to determine the region containing $S$. Using the information of faces about edges (radially ordered), this can be done in time proportional to the number of faces around an edge (which generally is also small).

However, it should be noted that if the simple patches are strongly linked this process needs to be executed only for the first patch $S_i$. The next patches to be processed are those that share an edge with $S_i$. In this way, one knows a priori one edge of the CGC identical to an edge of each new patch. This can be exploited to accelerate the search for vertices of the CGC identical to vertices of the new patch. The region containing the new patch can be readily determined and the search for identical vertices can be restricted, then, to those vertices that are in that region.

If the patches are not simple, the complexity of the creation algorithm depends on the complexity of the geometrical algorithms used in determining face intersection. This complexity depends on face geometry. In any case, the processing time depends on the number of face pairs to be checked for intersection. If $F$ is the final number of faces, then it is possible to construct the CGC checking $O(F^2)$ pair of faces for intersection. Of course, this is not the most efficient way.

To improve the insertion time, we propose a recursive algorithm that exploits the adjacency of the faces created when a new patch $S$ is inserted. If a region $R$ contains some point of $S$ and if $S$ crosses a face of the CGC, then at least one of the faces crossed is on the boundary of $R$. Thus, if $S$ does not cross any face of $R$, then $S$ is entirely contained in $R$. Otherwise, $S$ must be refined along the intersection found. For each of these pieces, a region containing at least some portion of it is readily available through adjacency informations. Therefore, the following algorithm can be used to insert $S$, assuming that it has a single face $f$:

1) Determine the region $R$ of the CGC that contains an arbitrarily chosen point on $f$.
2) Refine the faces on the boundary of $R$ and $f$. Put $f$ and each new face created in $S$ in a list $L$.
3) Take a point on each face in $L$ and check its inclusion in $R$. If the point is in $R$, then the corresponding face is entirely contained in $R$. In this case, the face is marked, removed from $L$ and put in a list $M$.
4) For each face in $M$, traverse its edges. If a given edge is adjacent to an unmarked face $g$ of $S$, then get the region $Q$ (which is adjacent to $R$ along that edge) and go to step 2 recursively, with $g$ playing the role of $f$ and $Q$ playing the role of $R$.

This algorithm only checks intersections with faces that may really be crossed. The most time consuming step is the first one, which takes time $O(F)$. The same algorithm can be used to combine two CGCs. The first one is considered as a CGC and the other as a set of linked faces. Now $S$ contains a set of linked faces and any of them can be chosen to start the process. Thus, the first step of the algorithm is executed just once for each strongly linked set of faces.

## *4.4 Generic Geometrical Handling*

Our implementation methodology uses three software layers with well defined functionalities: one layer maintains topology (this layer completely ignores geometry); another layer manages and manipulates geometrical entities; and a third layer handles user interaction.

Conceptually, the system is structured to handle arbitrary curves and surfaces. In order to achieve that, geometry is dealt with through a standard set of functions. Whenever a new geometry type is introduced, it suffices to provide its specific version of the set of standard functions, which are described below.

1) Given a pair of cells of the same dimension, check if they are geometrically identical.
2) Given two faces, check if they intersect each other and return a list with vertices and edges to be created in each face.
3) Given a cell, return the coordinates of one of its interior points.
4) Given a cell and the coordinates of a point, check for point inclusion in the cell.
5) Given a cell, return a measure of its size. The measure of an edge is its length; that of a face is its area; and that of a region is its volume.
6) Given an edge $e$ of a CGC and a face $F$ to be inserted in the CGC, with $e$ as one of its edges, return the face that succeeds $F$ in the ordered cycle of faces about $e$.
7) Given a pair of cells $C_1$ and $C_2$, with the same dimension, produced as a consequence of the division of a cell $C$, distribute the geometrical attributes of $C$ between $C_1$ and $C_2$.
8) Given a pair of cells $C_1$ and $C_2$, with the same extent, and a cell $c$ on the intersection of the boundaries of $C_1$ and $C_2$, combine the geometrical attributes of $C_1$ and $C_2$ to produce the geometrical attributes of the cell, which results when $c$ is deleted.
9) Given a cell, allocate or free the memory area containing its attributes.
10) Given a file and a cell, write the cell attributes to the file and return the number of records written.
11) Given a file and a number indicating the quantity of records to be read, get the attributes of the cell from the file and return a pointer to the area allocated for them.
12) Given a cell, draw it.

Functions 9–11 are used to store a CGC into permanent storage and to retrieve it back. Function 12 is used to display a CGC on a graphic device. Depending on geometry, some of these functions (e.g., Function 2) may not be easily implementable. Approximation methods should be used in these cases.

## *4.5 Application Example: Discrete Element Modeling*

The Discrete Element Method is a numerical method that geotechnical engineers use to analyze the interaction and movement of rock blocks originated from natural fractures in a rock mass [24]. The basic idea of this method is to determine the equilibrium of each block, taking into account distributed body forces, such as its own weight, and contact forces with adjacent blocks along the fracture joints. Hydraulic pressure of fluid flow through the joints can also be considered.

The Discrete Element Method is a powerful numerical simulation tool which can handle problems of arbitrary geometry and shape. From the modeling perspective, the method is simpler to use than the Finite Element Method [25] because discrete elements have no pre-specified topology, as required by finite elements. However, discrete element modeling, like finite element modeling, is still an open issue: model creation and manipulation during simulation require several sophisticated modeling procedures. For example, knowing the adjacency relationships among the several blocks, including an explicit knowledge of the contact areas (lines), permits an efficient simulation of this phenomenon. Study of water percolation can also be made much more efficient if the adjacency information among the blocks and their interfaces is available.

In two dimensions, the joints are just lines and the adopted modeling process in the construction of two dimensional discrete element models is based on the insertion of curve segments (defined, say, with a digitizing tablet). This process may be adopted in any system based on a planar subdivision representation.

A similar modeling procedure can be used in three dimensions: the incremental procedure for creating spatial subdivisions described in the previous section is a natural approach for modeling solid discrete elements. The model can be generated inserting families of joints planes (rock fractures), each family defined by an orientation and a sequence of joint positions in space. An example of the three dimensional modeling procedure is shown in Figures 7 to 10. Fig. 7 shows patches of fracture joints, which are trimmed outside the modeled portion of the rock mass. A solid view of the model is shown in Fig. 8. Distinct lithology properties are assigned to different rock blocks. The complete spatial subdivision data representation scheme, augmented by appropriate attributes, provides information required for a solid discrete element analysis. An input data file suitable for analysis is extracted from the radial-edge description to feed a discrete element program.

The proposed data representation is also very helpful for model visualization [26]. In Fig. 9, some of the blocks were "turned off," i.e., made invisible. This effect is accomplished simply by not displaying the faces on the boundaries

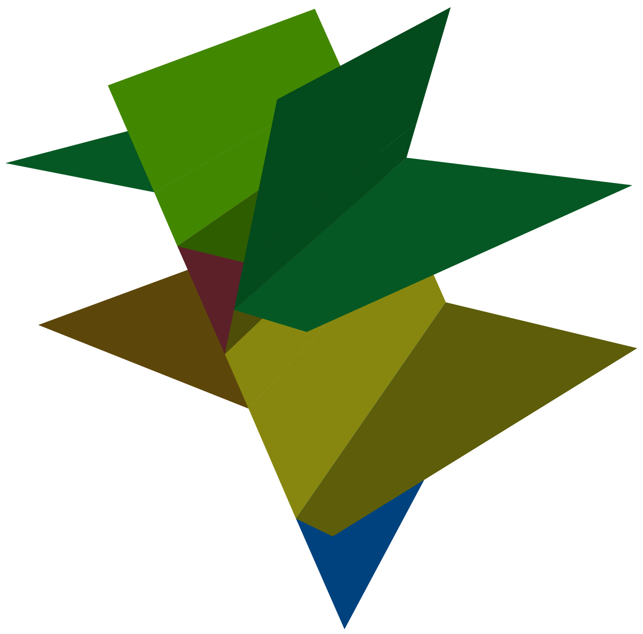

Fig. 7. Patches of joints in three dimensions.

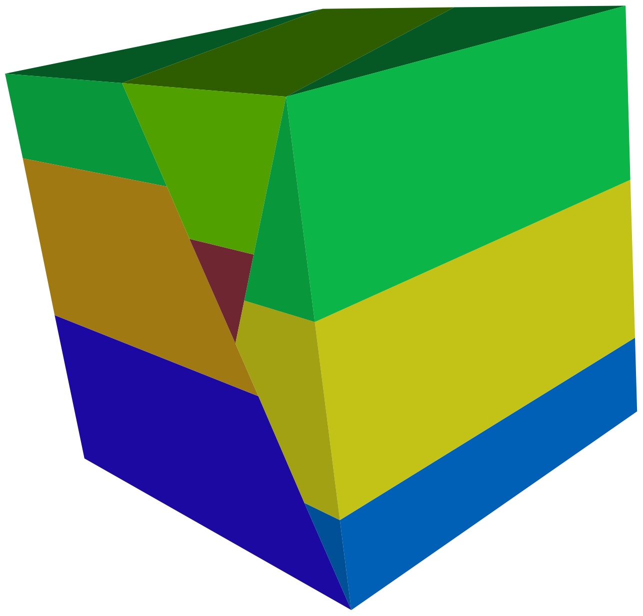

Fig. 8. Solid visualization of three dimensional discrete element model.

of the invisible regions. Another effective strategy adopted for model visualization is shown in Fig. 10. This figure shows the effect of a cutting plane, parallel to the screen plane, that cuts a corner of the modeled portion of the rock mass, also clipping its image. The central point of this visualization strategy is to suggest the user a solidity property for the discrete element model. Although the model is represented in the data structure as a set of

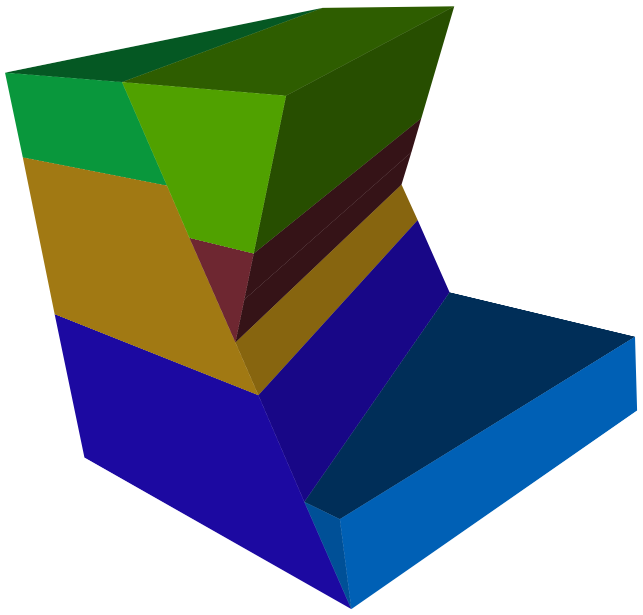

Fig. 9. Solid visualization with some blocks made invisible.

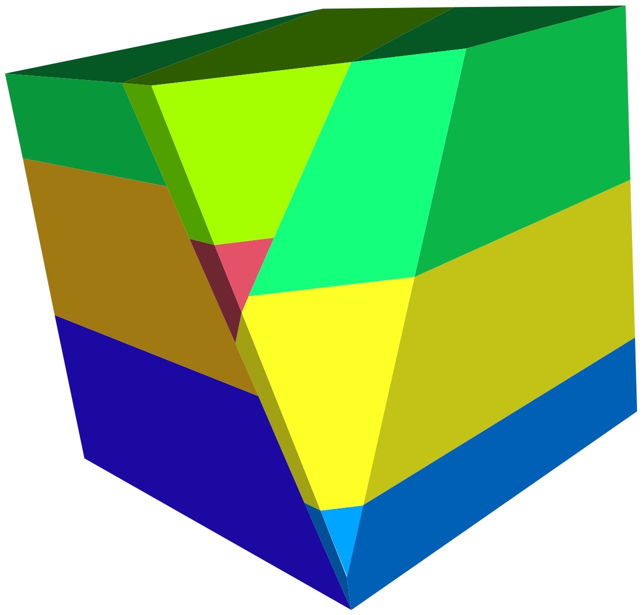

Fig. 10. Cutting of discrete element model.

solid regions, it is always exhibited on the screen by displaying the surfaces on the boundaries of the regions. The strategy accomplishes its goal because it automatically creates, in real time, fictitious faces at the cutting plane. The colors of the corresponding cut regions are assigned to these faces. The solidity sensation results from the display of the fictitious cutting faces in addition to the non-clipped surfaces.

The same procedure described previously to insert a new surface patch in a spatial subdivision is used for the creation of the fictitious cutting faces [26]. This procedure is fast because it exploits all the adjacency information provided by the data structure.

This methodology also provides topological and geometrical support for interactive generation of finite element meshes. The same topological representation adopted in this work has been successfully used in conjunction with meshing algorithms in engineering applications, such as for simulation of fracture propagation in 3D [27] and of 3D reinforced concrete subassemblages [28].

## 5 Construction of Heterogeneous Objects

As mentioned before, we have chosen CGC as the mathematical model to describe a spatial subdivision. This approach provides a basis for representing complex objects in two steps: *subdivision* followed by *selection*. In the first step, space is subdivided in a special hierarchical way: the entire space is subdivided into regions (three dimensional cells), whose boundaries are subdivided into faces (two dimensional cells), whose boundaries are subdivided into edges (one dimensional cells), whose boundaries are formed by vertices (zero dimensional cells). In the second step, the cells whose points correspond to the object (the *active* cells) are selected. The resulting object (a complete geometric complex together with a set of active cells) is called a SCGC (*Selective Complete Geometric Complex*). This terminology is borrowed from Rossignac and O'Connor [17], who have introduced SGCs (*Selective Geometric Complexes*). The only difference is that we require geometrical complexes to be complete.

Fig. 11 shows an object consisting of a bar of a material $A$ (e.g., steel) inserted into another bar of a material $B$ (e.g., concrete). In order to represent this object by a SCGC, we first subdivide space in three regions $R_1$, $R_2$ and $R_3$ and the faces, edges and vertices determined by them. Next, regions $R_1$ and $R_2$ are associated with materials $A$ and $B$, respectively, while the external region $R_3$ remains inactive (we also suggest using a *neutral* material, which is associated with the cells which constitute the interface of these regions). In this example, all cells of a given dimension are contained in an affine space of the same dimension (all faces are planar and all edges are straight). The SCGC model includes more general structures, in which the cells of a given dimension are only required to be contained in an algebraic variety of that dimension.

Our goal now is to propose a process to construct heterogeneous objects that is adequate to interactive applications. The proposed construction process (or modeling process) is based on a family of operators, which act on space

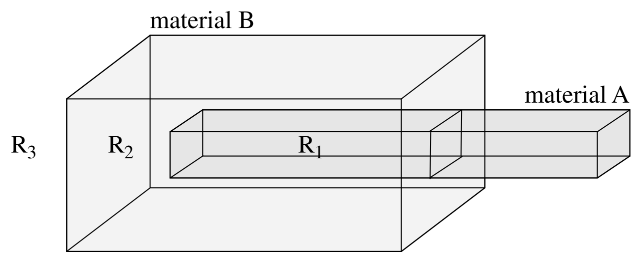

Fig. 11. An object and the space decomposition associated with it.

subdivisions. Before describing these operators, we briefly discuss modeling processes in general and describe the role they play in interactive modeling systems.

### *5.1 Modeling Processes*

A *modeling process* is a process in which one starts from an initial model $M_0$, and obtains intermediate models $M_1, M_2, ..., M_{n-1}$ through successive refinements, until a (satisfactory) final model $M_n$ is obtained. Ideally, $M_0$ should be as close to $M_n$ as possible and the refining operations should be as powerful as possible, in order to minimize the number of intermediate models. An important question is: what is the best way to obtain, interactively, each model from its predecessor?

In [23] we give a modeling process which is adequate for two dimensional applications. There, a planar subdivision was constructed from curve segments (defined, say, with a digitizing tablet). However, the analogous process is not satisfactory to build three dimensional space subdivisions: most people are not willing to build a three dimensional object by specifying each surface patch on its boundary.

The need for a modeling process more efficient than the direct specification of the bounding surface of a solid has long been recognized in the field of Geometric Modeling. Most BRep based representation systems provide modeling tools to release the user from the task of directly specifying each bounding surface.

The CSG (*Constructive Solid Geometry*) modeling process [2] is widely used in solid modeling because it provides sculpting refining mechanisms which are familiar to most people. Through regularized boolean operations, certain basic solids are combined to yield increasingly complex objects. In certain modeling systems, the only representation kept for an object is the sequence of constructions used to build the object (the CSG tree). In these systems, there is no explicit boundary evaluation. In other systems, the CSG method is used only to define objects. Once an object is defined, its CSG tree is evaluated to provide a boundary representation.

The CSG modeling process does not work, however, to build general SCGCs, since CSG boolean operations are defined only over regularized (i.e., dimension-homogeneous) solids. A more general process was introduced by Rossignac and Requicha [15]. They introduced a new class of objects, called CNRG (*Constructive Non-regularized Geometry*) objects, and a family of operators which act on these objects. Space subdivisions corresponding to CNRG objects can be implicitly represented by the corresponding tree, in the same way as CSG trees can be used to represent solids. In this paper, however, we are interested in using CNRG operators only in the modeling process. These operators provide a convenient way to define topological and boolean operations for SCGCs.

## *5.2 Constructive Non-Regularized Geometry*

A CNRG object is a collection of mutually disjoint (possibly disconnected) components. A component is a (possibly non regular) subset of $\Re^n$. The pointset $pA$ corresponding to a CNRG object $A$ is the union of the pointsets corresponding to each of its components.

A CNRG tree is a rooted directed acyclic graph and represents an object. The leaves are the CNRG primitives. Internal nodes represent intermediate CNRG objects obtained by applying certain operators to their children. The operators are: aggregation, unification, sum, product, subtraction, complementation, interior, closure, boundary and regularization. Different symbols are used in order to distinguish these operators from the corresponding pointset operators. The symbols $*, +, -, i, c, k, b, r$ are used for the CNRG product, sum, difference, interior, complement, closure, boundary and regularization.

CNRG operators provide a modeling process suitable for user interaction. When followed by a unification operator, which creates an object having only one component, they behave exactly as the corresponding pointset operators. Without unification, however, they return an aggregate of pairwise disjoint components.

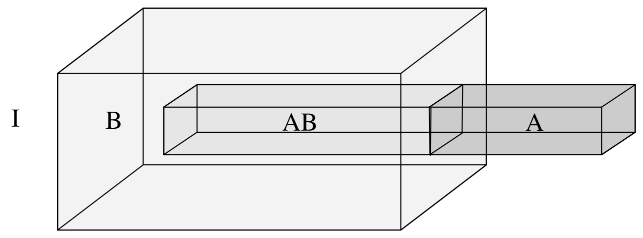

Fig. 12. Combining SCGCs.

### *5.3 CNRG Combination of SCGCs*

The definition of a CNRG is quite general: it is only required that its components be pairwise disjoint. However, particularly interesting CNRGs are obtained from SCGCs. Indeed, a SCGC can be seen as a CNRG whose components are its active cells. This allows one to use CNRG operators to build and modify SCGCs. In fact, one of the main proposals of this paper is to employ the CNRG modeling process using SCGCs as the underlying representation scheme. That is, we describe how to implement unary and binary CNRG operators which act on SCGCs. The initial step for the binary operators is to *combine* the respective SCGCs, using the methodology described in Section 4. Combining two SCGCs $A$ and $B$ means obtaining another SCGC $C$, compatible with both $A$ and $B$. This means that $C$ must be a common refinement of $A$ and $B$. That is, each cell of $C$ must be contained in some cell of $A$ and some cell of $B$ (these are called the *origin* cells). Moreover, each cell of the resulting complex $C$ is assigned an *origin* attribute, which can take one of the following four values: I, A, B or AB. The value I indicates that both origin cells are inactive, the value A (B) indicates that only the origin cell in $A$ ($B$) is active and AB indicates that both cells are active. Fig. 12 shows two simple SCGCs (corresponding to the bars in the example of Fig. 11) and the SCGC resulting from their combination.

Once the cells of the combined complex $C$ are classified according to their origin, the result of the CNRG operation can be immediately obtained by activating the proper cells, as will be described in this section.

Therefore, the crucial step in implementing CNRG operators for SCGCs is combining the SCGCs, which consists in obtaining a common refinement and then classifying its cells according to their origin.

To perform the combination of complexes, the skeletons of $A$ and $B$ (that is, the set of their faces and edges) must be compatible. This can be achieved by using an extension of the procedure described in Section 4. We find all

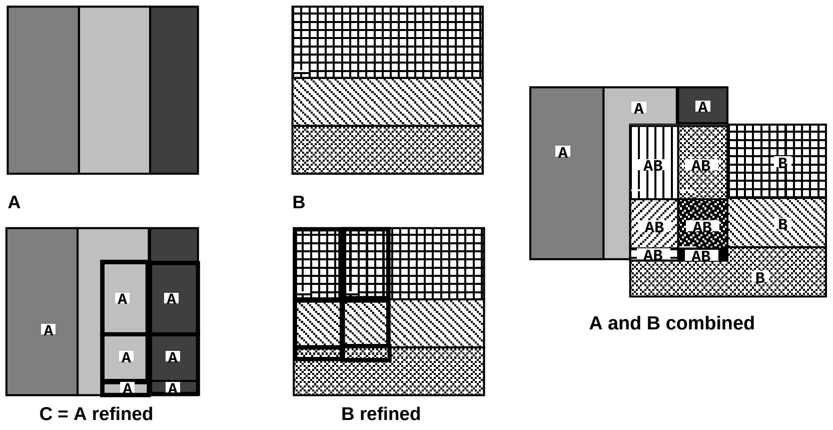

Fig. 13. Combination of SCGCs.

intersections between faces and edges of $A$ with faces and edges of $B$; each face or edge involved in one of these intersections is appropriately refined. Finally, the already refined faces and edges of $B$ are stitched into the refined complex $A$. When each cell is added, its origin attributes and those of the regions that are affected by its inclusion are appropriately established.

The steps to be executed are detailed below and illustrated in Fig. 13.

- Refine $A$ and $B$, in such a way that they "fit" each other.
- Create a complex $C$, identical to the refined version of $A$. Assign the value A to each active cell of $C$ and the value I to its inactive cells.
- Add to $C$ each facet $F$ (face together with its boundary) of $B$, as described in Section 4. Different actions are called for, depending on whether a given cell of $B$ already exists in $C$ and depending on whether that cell is active or inactive in each complex. Active cells of $B$ that already exist and are active in $A$ receive attribute AB. Active cells that do not exist or are inactive in $C$ are assigned attribute B. All regions created in this process are initially inactive.
- Traverse each region of $C$ and check whether it is contained in some active region of $A$ or $B$, accordingly updating its `origin` attribute. This can be done by resorting only to adjacency information; no geometric algorithm is required.

The procedure above classifies each cell of the combined complex $C$ according to its `origin`. Then, the result of the desired CNRG operation is obtained by selecting the appropriate cells.

- Sum: all cells having `origin` different from I are activated.
- Product: only cells having `origin` AB are activated.
- Subtraction $(A - B)$: cells having `origin` A are activated.

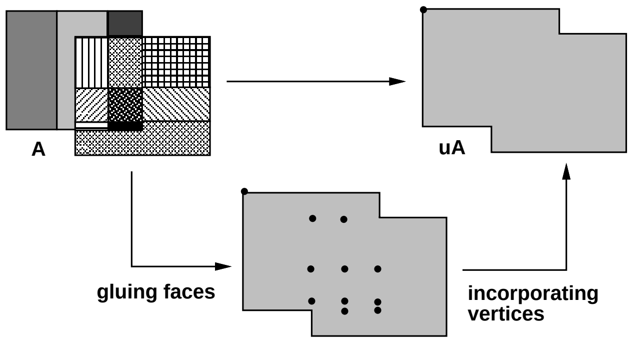

Fig. 14. Simplification of a CNRG object.

Implementing unary CNRG operators is simpler, since they act on only one complex $A$. Each operator only needs to make active or inactive the appropriate cells of $A$.

- Interior: for each inactive cell $c$, the cells which are on the boundary of $c$ must be rendered inactive.
- Closure: for each active cell $c$, the cells which are on the boundary of $c$ must be activated.
- Boundary: is the set of all cells which are on the closure but not in the interior of $A$.
- Regularization: is obtained by taking the interior of $A$ and then the closure of the resulting complex.

Therefore, we have shown that any system capable of representing complete SCGCs can easily be adapted to perform unary CNRG operations. It should be stressed, however, that one unary CNRG operator (the unification operator) is missing from the list above. This operator, at least in its original form, cannot be applied to SCGCs, since the union of all its active cells is not, in general, a valid cell. Instead of the unification operator, one must employ the *simplification* operator for SCGCs. This operator is described in [23] and is implemented through the *elimination* of inactive cells, which are not needed for the model; *gluing* of active cells having the same geometrical support and such that their common boundary is also active; and *incorporation* of active cells immersed in another active cell. Fig. 14 illustrates the simplification process for a SCGC.

The CNRG operations provide a powerful set of modeling tools for creating and modifying SCGCs. To implement such operations it suffices to be able to make SCGCs compatible. Once those are made compatible, all operators are reduced to making cells active or inactive.

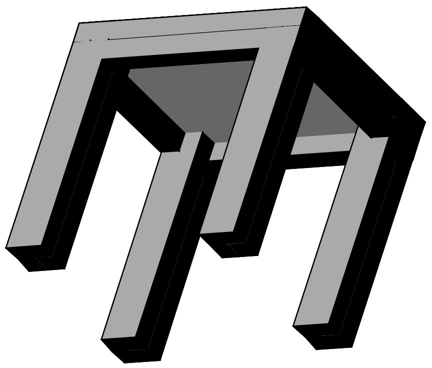

Fig. 15. CSG union.

## *5.4 Modeling Heterogeneous Objects*

The CNRG operators are especially useful in modeling heterogeneous objects, that is, objects composed by several materials. Let us consider, once more, the object shown in Fig. 11, constituted by a steel bar partially inserted in a concrete bar. As we have seen, this object is naturally represented by a SCGC having three regions, corresponding, respectively, to the steel bar, the concrete part and the unbounded external region. Moreover, this object is easily described using CNRG operators. If $B$ denotes the concrete bar (before the insertion of the steel bar) and $A$ denotes the steel bar, then the final object is given by $(A - B) + B$.

Another example is given in Fig. 15, which shows a structure constituted by four concrete columns, four steel beams and one slab, conveniently positioned in space.

This structure can be created by the CSG union operation; however, the resulting object would have just one region (Fig. 15). By using the CNRG sum operation, one obtains 27 distinct regions (Fig. 16). Each of these regions must be associated with a single material. A natural question is: which material should be associated with the regions of the resulting object corresponding to the intersection of regions associated with different materials in the original parts?

A possible scheme to automatically determine the material to be attributed to each region consists in assigning to each material a number, called *dominance factor*, which establishes the material which predominates in an intersection operation. For instance, if the dominance factor of steel is 5 and that of concrete is 3, the regions originated by intersecting steel and concrete components will be considered to be constituted by steel.

One should observe that, even if the desired result is a structure having just one

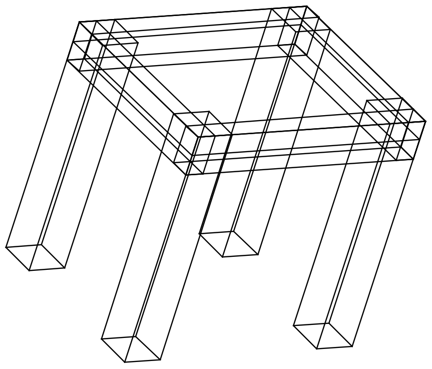

Fig. 16. CNRG sum.

region, such a structure can be obtained by the simplification operation, which consists in eliminating the faces that are not used by the external region and the edges (vertices) which separate faces (edges) having the same geometrical support.

## *5.5 Adding CNRG Operators to BRep Modeling Systems*

An important fact about the previously described operators is that they can be added to BRep modeling systems in order to increase their modeling power. This is done through two procedures. The first one reads the description of the boundary of a solid represented by such a modeling system (which is necessarily a two-manifold) and creates the corresponding two region SCGC. The second one does the opposite.

After obtaining the representation of an object through appropriate CNRG operators, one can traverse its regions (excluding the external one) and make each into a closed three-manifold (this is done by removing dangling faces, dangling edges and point shells). Now, each shell is a two-manifold and can be represented in any BRep (manifold) modeling system. Therefore, the object which results from the CNRG operators can be described as a disjoint union of distinct two-manifold solids, which can be returned to the BRep modeling system. Such an extension was added by us to the GeneSys modeling system [29].

The model in Fig. 11, for instance, can be produced by GeneSys. First, the two-manifold solids corresponding to each bar are created in GeneSys. Then, SCGCs $A$ e $B$ are created from the descriptions of these solids. Next, the CNRG expression $(A-B)+B$ is evaluated, yielding a new SCGC $C$ which has three regions. Two of these regions are active and are constituted by different materials. Finally, each of these regions is returned to GeneSys, originating two separate solids. Geometrically, the resulting object (which is the aggregate of the two solids) is the same as the one represented by the SCGC $C$. However, all adjacency information between the two parts is lost in the conversion.

## 6 Conclusions

The main contribution of this work is a new methodology for dealing with subdivisions of the three dimensional space. This methodology was used for implementing a system capable of generating geometrical and topological descriptions of a spatial subdivision from a set of surface patches. The system provides a set of functions that can be called by an application program for manipulating CGCs.

The techniques described here can also be used to model objects made of different materials. One can associate to each topological element (region, face, edge or vertex), an attribute representing the corresponding material. For this application, we actually need to model aggregate of objects, i.e., solids, possibly combined with lower dimensional parts. However, the mechanism of creation of a CGC from surface patches is not very natural. Rossignac and Requicha described the CNRG representation scheme [15] to define set operations analogous to CSG operations that can be applied to aggregates of objects. We have shown here that CNRG operators can naturally be implemented in a system based on complete spatial subdivisions, thus providing a representation scheme maintaining an explicit representation of cells forming a non homogeneous object and, at the same time, allowing its constructive creation from CSG-like operations. This modeling technique — producing more complex CGCs from simpler ones through CNRG operators — can be thought of as an extension of boolean operators used in manifold modeling [1] or in nonmanifold modeling [13,14].

The main topic for future research in this area is the search for more efficient algorithms to combine SCGCs. Adjacency information from each SCGC is exploited for finding the intersection of their skeletons. However, the inclusion of faces and edges in the new model is done individually. Better use of the adjacency information may lead to more efficient combining algorithms.

## Acknowledgement

The authors acknowledge the technical support of PUC-Rio (TeCGraf) and IMPA (VisGraf Project), without which this work would not be possible. TeCGraf is partially funded by CENPES (Petrobrás Research Center) and CEPEL (Eletrobrás Research Center), both located in Rio de Janeiro, Brazil. IMPA is a research institute of CNPq an agency of the Brazilian government. We also had a grant from CNPq GEOTEC-PROTEM project and from FAPERJ for the participation in the IV SIAM conference on CAGD.

The authors owe a debt of gratitude to Prof. Marcelo Gattass and Prof. Jonas Gomes for providing the adequate environment in TeCGraf and VisGraf, respectively, and to Luiz Henrique de Figueiredo for supplying many valuable suggestions to the authors. Figures 7 to 10 were produced by an interactive graphics program developed by Beatriz Castier during her MSc Program in Computer Science at PUC-Rio.

## References

[1] Martti Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, Maryland, 1988.

[2] Ari Requicha. Constructive solid geometry. Technical Memo. 25, University of Rochester, Production Automation Project, November 1977.

[3] James R. Miller. Architetural issues in solid modelers. *IEEE Computer Graphics and Applications*, 9(5):72–87, September 1989.

[4] Ari Requicha and H. B. Voelcker. Boolean operations in solid modeling: boundary evaluation and merging algorithms. *Proc IEEE*, 73(1):30–44, 1985.

[5] Vadim Shapiro and Donald L. Vossler. Efficient CSG representation of two-dimensional solids. *Transactions of the ASME Journal of Mechanical Design*, 113:292–305, September 1991.

[6] Vadim Shapiro and Donald L. Vossler. Construction and optimization of CSG representations. *Computer Aided Design*, 23(1):4–20, January/February 1991.

[7] Vadim Shapiro and Donald L. Vossler. Separation for boundary to CSG conversion. *ACM Transactions on Graphics*, 12(1):33–55, January 1993.

[8] Leonard Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[9] D. P. Dobkins and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of the Third ACM Symposium on Computational Geometry*, pages 86–99, Waterloo, Canada, June 1987.

[10] M. J. Laszlo. *A Data Structure for Manipulating Three-dimensional Subdivisions.* PhD thesis, Department of Computer Science, Princeton University, August 1987.

[11] P. Lienhardt. Extension of the notion of map and subdivisions of a three-dimensional space. In *STACS'88 Proceedings of the Cinquième Symposium sur les Aspects Thèoriques de L'Informatique*, Bordeaux, France, February 1988.

[12] Eric Brisson. *Representation of d-dimensional Geometric Objects.* PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, USA, 1990.

[13] Christoph Hoffmann. *Geometric and Solid Modeling: an Introduction.* Morgan and Kaufmann Publishers, 1989.

[14] Gary A. Crocker and William F. Reinke. An editable nonmanifold boundary representation. *IEEE Computer Graphics and Applications*, 11(2):39–51, March 1991.

[15] Jarek R. Rossignac and Ari G. Requicha. Constructive non-regularized geometry. *Computer Aided Design*, 23(1):21–32, 1991.

[16] Paulo Cezar Pinto Carvalho, Marcelo Gattass, and Luiz Fernando Martha. A software tool which allows interactive creation of planar subdivisions, and applications to educational programs. In E. Oñate, editor, *CATS'90 Proceedings of International Conference on Computer Aided Training in Science and Technology*, pages 201–207, Barcelona, Spain, July 1990. CIMNE Pineridge Press.

[17] Jarek R. Rossignac and Michael A. O'Connor. A dimension-independent model for pointsets with internal structures and incomplete boundaries. In *Geometric Modeling for Product Engineering*, pages 145–180. North-Holland, 1990.

[18] Kevin Weiler. The radial-edge structure: A topological representation for non-manifold geometric boundary representations. In *Geometric Modeling for CAD Applications*, pages 3–36. North-Holland, 1988.

[19] Kevin Weiler. *Topological Structures for Geometric Modeling.* PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, August 1986.

[20] S. Murabata and M. Higashi. Non-manifold geometric modeling for set operations and surface operations. Technical memo., Rensselaer Polytechnic Institute, June 1990.

[21] Eric Haines. Point in polygon strategies. In Paul Heckbert, editor, *Graphics Gems IV*, pages 24–46. Academic Press, Boston, 1994.

[22] Paulo Cezar Pinto Carvalho and Paulo Roma Cavalcanti. Point in polyhedron testing using spherical polygons. In Alan Paeth, editor, *Graphics Gems V*, pages 42–49. Academic Press, 1995.

[23] Paulo Roma Cavalcanti. *Creation and Management of Space Subdivisions.* PhD thesis, Department of Informatics, PUC-Rio, Rio de Janeiro, Brazil, March 1992. Writen in Portuguese.

[24] P. A. Cundall. Rational design of tunnel supports: A computer model for rock mass behavior using interactive graphics for input and output of geometrical data. Technical report, University of Minnessota, Mineapolis, Minnessota, 1974.

[25] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method Volume 1: Basic Formulation and Linear Problems.* McGraw-Hill, fourth edition, 1989.

[26] Beatriz Castier. Visualization of 3d geological formations represented by spatial subdivisions. Master's thesis, Department of Informatics, PUC-Rio, Rio de Janeiro, Brazil, April 1995. Writen in Portuguese.

[27] Luiz Fernando Martha. *Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three Dimensions.* PhD thesis, Cornell University, Ithaca, N.Y., 1989.

[28] David Potyondy, John Abel, and Anthony Ingraffea. An interactive environment for the simulation of 3d reinforced concrete subassemblages. In *Computer Aided Analysis and Design of Concrete Structures: Proc of SCI-C 1990*, pages 503–514, Swansea, U.K., 1990. Pineridge Press.

[29] Rolf Fischer. Genesys - hybrid system for solid modeling. Master's thesis, Department of Informatics, PUC-Rio, Rio de Janeiro, Brazil, August 1991. Writen in Portuguese.