An OOP Approach for Mesh Generation of Multi-Region Models with NURBS

William M. Lira[1] , Paulo Roma Cavalcanti[2] , Luiz C. G. Coelho[1], Luiz F. Martha[1]

[1]Department of Civil Engineering and
Computer Graphics Technology Group (Tecgraf),
Pontifical Catholic University of Rio de Janeiro (PUC-Rio) 22453-900,
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ, Brazil.
{william, lula, lfm}@tecgraf.puc-rio.br
[2]Department of Computer Science,
Federal University of Rio de Janeiro (UFRJ) 21945-970,
Cidade Universitária, Ilha do Fundão, Rio de Janeiro, RJ, Brazil.
roma@lcg.ufrj.br

Abstract: This paper presents an object-oriented approach for creating multi-region non-manifold models with NURBS. The main motivation is that the geometry and shape of realistic engineering objects are intrinsically complex, usually composed by several materials and regions. Therefore, automatic and/or adaptive meshing algorithms have become revealed themselves quite useful to increase the reliability of the procedures of a FEM numerical analysis. The present approach is concerned with two aspects of 3D FEM simulation: geometric modeling, with automatic multi-region detection, and support to automatic finite element mesh generation. The final objective is to use geometric models directly in numerical applications.

1 Introduction

Finite element analysis [1,2] and geometric modeling of solids [3,7] are important items in the process of simulating engineering problems (see, for example, Figure 1), especially when an analytic solution is unknown or is difficult to obtain.

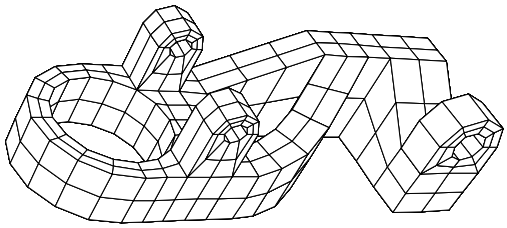

Figure 1: A geometric model and its finite element mesh.

Generally, the Finite Element Method (FEM) is based on a numerical model obtained from the refinement or "discretization" of the problem domain, combined with additional information necessary for the complete definition of the physical problem. Such information consists of a set of parameters, called simulation attributes [4,5,6]. The discretization, denominated finite element mesh, consists of a group of nodes or vertices (points with coordinates) and a group of cells, called finite elements, with a predefined topology (triangular, quadrilateral, or tetrahedral, for example). The elements are defined by a list of node connectivities (sequence of vertices that belong to each element). A finite element model is the association of a finite element mesh with a set of simulation attributes.

One important aspect of 3D FEM simulation is mesh generation. This is an area that has been active since the creation of the method. In general, the mesh generation process demands time and is quite tiresome, apart from demanding a certain degree of experience from the responsible professional for this task. In this context, automatic and/or adaptive meshing algorithms have become revealed themselves quite useful to increase the reliability of the procedures of a FEM numerical analysis [1].

Another important aspect in three-dimensional finite element simulation is the creation of the geometric model. There are several issues involved in this task, ranging from user-interface strategies to data representation schemes. To accomplish this, it is necessary to use special programs, called modelers, which can digitally reproduce common geometric forms of the target objects [7]. The geometry and shape of realistic engineering objects are intrinsically complex, usually composed by several materials and regions.

The modeling methodology discussed in this paper is concerned with two aspects of 3D FEM simulation:

- geometric modeling, with automatic multi-region detection, and
- support to automatic finite element mesh generation.

This methodology was implemented in an existing finite element modeler called MG. This is a finite element pre-processor that was originally devised for the generation of surface (shell) finite element models [8]. Later on, the system was extended to also consider solid meshes [9,10]. MG's modeling capabilities address two important issues in 3D finite element modeling. The first is related to user-interface and interactive graphics procedures to

generate surface meshes [11], and the second is the intersection of surface finite element meshes, such as the ones shown in Figure 2 [12].

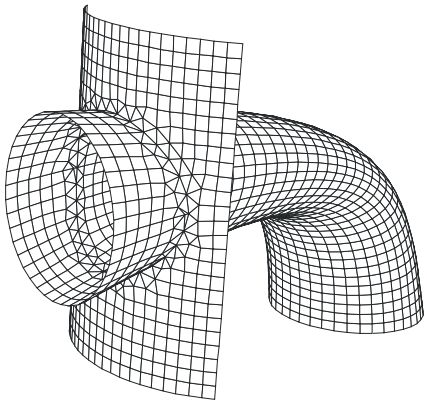

Figure 2: Intersection of surface meshes.

The original version of MG is powerful in model representation, and has a relatively simple and efficient interface. However, its data structure was not based on any formal geometric modeling concept. The geometric consistency of a model in many situations relies on user intervention. For example, there is no capability to automatically detect when a volume in space is enclosed off by a set of surface patches. The user must explicitly indicate this, which may be a hard task in a realistic engineering model. This capability is particularly important for finite element mesh generation, as, sometimes it is desirable to have several regions, each one meshed by a different algorithm.

The formalism necessary for modeling capabilities that allow automatic recognition of created solid regions goes beyond the scope of this paper. The methodology adopted here was previously devised by the authors [13], and is based on a complete topological representation of a space subdivision, called Complete Geometry Complex (CGC), which is summarized in the next section.

This paper describes a class organization, in the context of Object Oriented Programming (OOP), of a new version of the MG modeler that, still keeping the simple and efficient user-interface characteristics, provides capabilities for automatic region detection with surface patches represented by Non-Uniform Rational B-Splines (NURBS) [14]. To achieve this, a hybrid approach was adopted in which a full CGC representation of the model is not maintained all the time. Instead, the previous MG data structure [8] was extended so that a CGC model can be created at any moment when the user requires region detection or surface intersection.

The new class organization also provides support for automatic surface and solid mesh generation. It is not the objective here to describe meshing algorithms. It suffices to say that surface mesh generation is performed in the parametric space of each surface and that the provided algorithms in MG are described elsewhere [9,10,12,15].

2 Complete Geometric Complex Representations

In general, there are two main strategies for the representation of a three-dimensional geometric model: an implicit scheme and an explicit scheme. The most common implicit scheme is Constructive Solid Geometry (CSG) [16], in which the target object is reproduced by a set of Boolean operations applied to primitive objects. In the explicit scheme, the geometry of an object is defined by a set of surface patches, which may be created through an interactive graphics interface. However, the complete definition of the solid model requires combinatorial relationships among the several surface patches, which will result in the definition of the interior region, model boundaries and other topological information. This type of solid model representation is called Boundary Representation (B-REP) [3,7].

Traditional CSG and B-REP techniques apply to objects that decompose space into three parts: interior, exterior, and boundary. This class of objects is referred to as manifold objects, because their boundaries are two-manifold sets in three dimensions [3]. This means that the original techniques could not model multi-region objects.

Many applications in engineering need to model multi-region objects or objects that present other non-manifold features, such as dangling faces or edges. For this reason, many works in the literature have proposed non-manifold modeling schemes [13,17,18,19,20,21,22, 23].

The generation of a consistent non-manifold B-REP model from a set of surface patches is not a simple task and may involve surface intersection and region detection. Considering the user-interface problems related to this task, it is desirable that the creation of a B-REP model from a set of surface patches be automated. The ideal environment for the user would be to create these patches with no explicit relationship, except using already created geometry information for the generation of a new patch. The modeler would automatically generate surface intersection curves and topologically link these entities to the geometric definition.

A previous work by the present research group [13] proposed a non-manifold approach for modeling multi-region objects. A general methodology for creating and manipulating a spatial subdivision in cells of arbitrary shape and geometry was developed. A spatial subdivision may be created by means of inserting planar surface patches one by one, allowing the insertion of new patches in real time. The object resulting from this decomposition is classified as Complete Geometric Complex (CGC) because it is a special case of Geometric Complex [22] that

occupies the entire three-dimensional space (the unlimited outer region is also represented in the subdivision).

Several works have presented methods to represent spatial subdivisions. Rossignac and O'Connor [22] addressed the general problem of representing n-dimensional objects, possibly with internal structures. Some data structures used in non-manifold solid modeling [17,18,19,21] represent, in a general way, the adjacency relationships of three-dimensional objects not necessarily homogeneous in dimension. In the present implementation, the Radial-Edge data structure (RED), proposed by Weiler [20,21] is adopted.

This data structure is known as Radial-Edge because it explicitly stores the list of faces radially ordered around an edge (Figure 3). The Radial-Edge data structure was conceived for non-manifold modeling and Weiler has proved its completeness, which means that any adjacency relationship can be extracted from this representation.

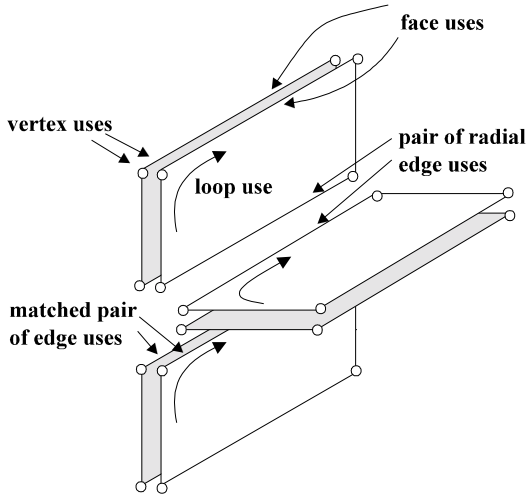

Figure 3: Uses of topological elements in RED.

In order to describe the topology of a spatial subdivision, the Radial-Edge representation employs the concept of *use* of a topological element. A use can be seen as the occurrence of a topological element in an adjacency relationship related to an element of higher dimension. Thus, the Radial-Edge structure 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 are composed by an alternating sequence of *edge-uses* and *vertex-uses* (Figure 3). *Vertex-uses* are necessary to store non-manifold conditions at vertices.

The RED structure is a hierarchical description of a spatial subdivision, starting in higher dimension levels (*regions*) and reaching the lower levels (*vertices*) (Figure 4). The topological elements are kept in doubly-linked circular lists and have pointers to their attributes.

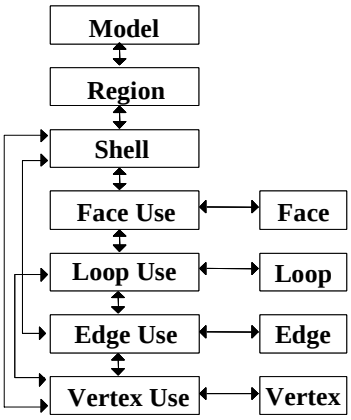

Figure 4: Hierarchy of topological entities in RED.

Topological data structures are complex and should not be manipulated directly. Weiler has introduced a set of operators that provide a high-level method to access the Radial-Edge structure. These operators are divided in two groups. The first group has operators that act on faces of a spatial subdivision and are analogous to the (two-manifold) operators presented by Mäntylä [7]. The second group has operators that are capable of creating wire-frames and of adding faces, which are attached to specified edges or wireframes. These are referred to as non-manifold operators. Considerations about a minimal set of operators can be found in [24].

The present CGC modeling is implemented as a library of OOP classes, which provide a set of high-level operators that manipulate a spatial subdivision. These operators receive as input geometric information on the surface patches that are inserted in the spatial subdivision. This geometric information is automatically translated into topological information required by Weiler's non-manifold and manifold operators.

The CGC modeling capability is one of the key aspects of the finite element modeling scheme proposed in this work. However, up to now the implementation of this methodology could only treat planar (polygonal) surface patches. One of the objectives of this work is to extend the methodology to consider curved surface geometries using NURBS.

3 Hybrid Modeling Approach

As previously mentioned, the proposed data representation scheme, which is adopted in the new version of the MG modeler, is based on a hybrid approach. The basic idea is to have two separate representations of the same model. One representation is stored in the modeler's data structure. This is the representation that is actually main-

tained in computer memory during the modeling process. The modeler's data structure is also permanently stored in disk when a model is saved. The other representation is a temporary conversion of the modeler's data structure into a CGC representation. In this conversion the model is modified to reflect possible surface intersection and is topologically treated to reflect region detection.

The main advantage of this approach is that it combines the functionality and simplicity of MG with the topological power of representation and robustness of CGC. There is a two-way communication between the two representations, as shown in Figure 5. CGC may be seen as a "topological engine" that generates topological information, that is consistent with the geometry of a model. The topological entities identified by the CGC representation are passed back to the MG representation, which includes regions that are automatically detected. In Figure 5, it can also be observed that the geometric description of the topological entities is common to both representations. This geometric description is stored in a separate module based on NURBS representation [25].

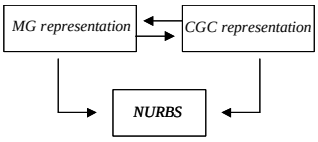

Figure 5: General modules in modeling approach.

The three modules shown in Figure 5 are implemented as OOP classes. CGC class organization is described in [26] and NURBS class organization is available in [25]. MG class organization will be described in the next section.

The basic information passed from the modeler's data structure to the CGC data structure consists of a set of surface patches defined by the user through MG's graphics interface. Each surface patch is inserted in the CGC representation, which might generate one or more topological *faces*. In instances where the patches only intersect each other at the boundaries, each surface patch will correspond to a *face* in CGC. Here, a surface patch is represented by its boundary curves and by its NURBS geometric description. These parameters are sufficient to determine a *face* in CGC module.

This module processes the surface patches passed by the MG module and generates a consistent topological model, which is then converted back to the MG representation. This conversion is performed, basically, by traversing all *regions* and *faces* generated by the CGC module and transforming them into entities of the MG representation. Each *face* in the CGC representation corresponds to a surface patch in the MG representation. Two patches may belong to a same geometric surface, in situations where an intersecting trimming curve subdivided the initial patch. Similarly, each edge in the CGC representation corresponds to a curve segment in the modeler's representation, and these segments may share the same geometric curve if they were originated from a curve split. Finally, each *region* in the CGC representation will generate a solid in the MG representation.

In the context of this work, the CGC representation is generated only when requested by the MG user. This means that consistency between geometry and topology is only enforced when desirable. The reason for this is that it might be very expensive to maintain this consistency after every step of the modeling process. Three-dimensional geometric modeling is a complex task that might involve a series of trial-and-error steps. The proposed hybrid approach was conceived focusing specifically on the problem. The MG representation is relatively simple and consequently provides an efficient manipulation of surfaces. It would not be possible to achieve the same degree of user-interface efficiency if, at any moment, consistency between topology and geometry were enforced after each user-interface task.

4 Modeler's Class Organization

The main data structure of MG was conceived with the purpose of supporting both surface (shells) and solid finite element modeling. The data structure class organization of MG incorporates the concept of geometric and topological entities. This structure also maintains the adjacency relationships among the entities. Figure 6 shows the OOP class organization in the MG module. The class diagram shown in this figure, as well as in other figures in this paper, follows the OMT (Object Modeling Technique) nomenclature [27].

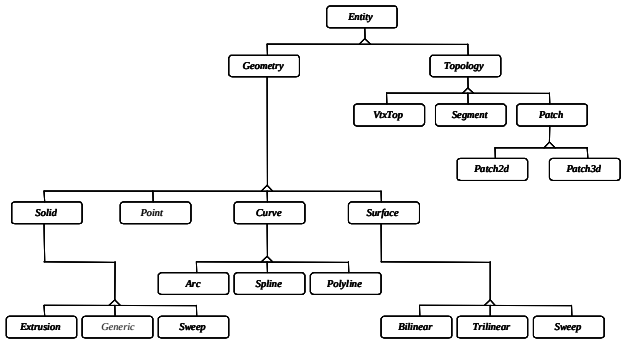

Figure 6: Modeler's general OOP class organization.

The *Entity* class is subdivided in two sub-classes. The first, *Geometry*, refers to the geometric entities of the model. The other, *Topology*, represents the entities that contain the topological information.

For example, in the MG data structure, a vertex in space has a topological version, which is represented by an object of the *VtxTop* class, and a geometric version, which is represented by an object of the *Point* class. A *VtxTop* object contains a list of references to adjacent curve segments (*Segment* objects) and a reference to the corresponding *Point* object. An object of the *Point* class stores the geometric position of a vertex in the 3D Euclidian space and has a reference to the corresponding *VtxTop* object.

Similarly, a *Segment* object represents the topology of a portion (segment) of a geometric curve. An object of this class stores the topological information (pointers to adjacent vertices, for example) and a reference to its geometric description (*Curve* object). Figure 7 illustrates the relationship between a curve (geometry) and its two segments (topology). In this case, curve *c1* has a reference to its geometric representation (a NURBS object) and a list of references to *Segment* objects that belong to it (*s1* and *s2*). These *Segment* objects contain references to their end vertices, (*v1*,*v3*) and (*v3*,*v2*), respectively, and to curve *c1*. A *Curve* object must have at least one *Segment* object.

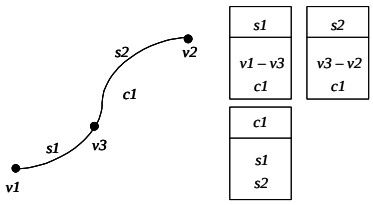

Figure 7: Relationship *Curve - Segment*.

Another important class in the context of the MG data structure is class *Patch2d*, which represents the topology of a portion of a surface. An object of the *Patch2d* class stores references to its boundary curves and to its surface geometric description, which is represented by an object of class *Surface*. A *Surface* object contains a reference to its geometric representation, i.e., a pointer to a NURBS object, and a list of references to *Patch2d* objects that belong to it. Figure 8 illustrates the relationship between surface *s1* and two corresponding *Patch2d* objects, *p1* and *p2*. Each *Patch2d* object contains topological information, which mainly consists of a list of *Segment* objects on its boundaries. In addition, a *Patch2d* object stores a surface finite element mesh that is eventually generated on it.

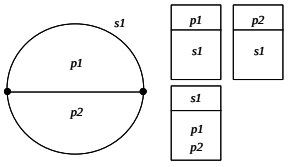

Figure 8: Relationship *Surface - Patch2d*.

An object of the *Surface* class must have at least one *Patch2d* object. A *Surface* object also contains geometric information related to the process that was used for its interactive generation. Currently, a surface may be generated from boundary curves by bilinear mapping, trilinear mapping, or sweep [8]. Therefore, each *Surface* object contains information about its creation method and a list of references to its generating curves (curves that were used for its creation).

A *Patch2d* object also has references to adjacent *Patch3d* objects. An object of the *Patch3d* class is the topological representation of a solid region. A *Patch3d* object has a list of references to *Patch2d* objects that form its boundary and a pointer to the corresponding *Solid* object. In addition, a *Patch3d* object stores a solid finite element mesh that is eventually generated in it.

The geometric version of the topological *Patch3d* class is the *Solid* class. A *Solid* object contains geometric information about the method used for its generation. Currently, there are three methods for solid generation: extrusion, sweep, or generic generation [9]. A generic generation of a solid may be performed explicitly by the user (by simply selecting surface patches on the boundary of the target solid) or automatically by the CGC representation when a solid region is detected. Therefore, each *Solid* object contains information about its creation method and a list of references to its generating curves and surfaces (curves and surfaces that were used for its creation).

An important issue addressed by this data representation scheme it the support for automatic and/or adaptive mesh generation. As previously mentioned, finite element mesh generation is an area of active research and is being treated by the authors elsewhere [9,10,12,15]. One of the goals of this research line is the development of a complete system for geometric modeling and 3D finite element adaptive simulation. The methodology for adaptive mesh generation has been tested in two-dimensions [28], and is based on the refinement of each topological entity in its own parametric space. For example, mesh generation of a surface patch (*Patch2d* object), which is performed in its parametric space, is based on a previous refinement of its boundary segments in their own parametric spaces. Similarly, solid mesh generation of a 3D region requires the previous refinement of its boundary surface patches.

The implementation of this mesh generation methodology is accomplished in the present data structure by means of the concept of *use* of a topological entity by a surface. The concept of *use* in the MG data structure is quite different than its concept in the Radial-Edge data structure. In this context, *use* simply means the geometric information of a certain entity in the parametric space of a surface. These *uses* are associated to the topological entities of the model. For example, a vertex that belongs to two adjacent surfaces has two *uses*. Figure 9 illustrates

this example. The single *use* of topological vertex *v1* refers to surface *s1*, with parametric coordinates $u = 0$ and $v = 0$. Topological vertex *v5* has two *uses*, one to *s1*, with parametric coordinates $u = 1$ and $v = 1$, and other to *s2*, with parametric coordinates $u = 1$ and $v = 0$. The same idea is adopted for segments. In this case, each use of a segment has the parametric coordinates of its end points.

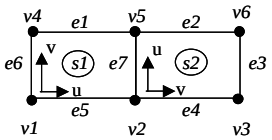

Figure 9: Uses of vertices in two adjacent surfaces.

Therefore, a *VtxTop* object has a list of *uses* associated to it in addition to the other topological information described previously. The *use* of a vertex is an object of the *PointUse* class. A *PointUse* object has a reference to its corresponding *Point* object and another reference to the corresponding *Surface* object. A *PointUse* object also contains two variables to store the parametric coordinates of a point on a surface.

Similarly, a *Segment* object has a list of *uses*, which are objects of class *SegmentUse*. A *SegmentUse* object has a reference to the corresponding *Segment* object and another reference to the corresponding *Surface* object. A *SegmentUse* object also contains variables to hold parametric coordinates of a curve segment on a surface.

5 Implementation of NURBS in the CGC Representation

The original CGC implementation treats only planar (polygonal) surface patches. However, one of the goals of this work is to consider CGC models with curved geometries, which is achieved by incorporating NURBS's geometry. Among other advantages, NURBS allows modeling more complex objects, possibly with holes, with arbitrary geometry, including conic surfaces.

The extension was relatively simple, because the CGC data structure does not depend on the underlying geometry, which means that it does not make any difference if an edge is considered as a straight line or a curve.

In the CGC representation, objects responsible for the definition of topological entities (*regions*, *faces*, *edges* and *vertices*) point to objects that describe their associated geometry and attributes (objects of classes *RegionAttr*, *FaceAttr*, *EdgeAttr*, *and VertexAttr*). In addition, CGC implements another class that concentrates all geometric algorithms. This is the *GeomPck* class which helps constructing a geometric model by calculating the volume of a certain region, the intersection between a surface and a curve, the location of a point in relation to a region, etc. These methods deal with geometric information associated to each topological entity. Therefore, modifications caused by the addition of a new geometry type in the CGC implementation are limited to a small group of classes that define and manipulate the geometry of a model.

FaceAttr and *EdgeAttr* objects hold the geometric description of *faces* and *edges*. This description is now stored in two new classes, as can be seen in Figure 10. The CGC *Surface* class defines the geometric representation of the surfaces of a model, while the CGC *Curve* class defines the geometry of the curves. Both *Surface* and *Curve* objects have their own parametric spaces. There are also the specific sub-classes for certain geometry types (for example, *Arc* class geometrically describes a circle arc and the *Gordon* class, a Gordon surface). The methods in these classes manipulate the corresponding geometric information. For example, there is a method that, given the parametric coordinates of a surface, obtains the 3D Euclidian coordinates of the corresponding point.

As previously mentioned, the geometric representation of curves and surfaces in the MG modeler uses a public domain OOP library [25] based on NURBS. This library provides the representation of several types of curves and surfaces, including the conic and quadric forms.

The connection between the MG/CGC representation and the NURBS library is simple. *Curve* and *Surface* objects contain references to their corresponding NURBS objects. Figure 11 illustrates this connection.

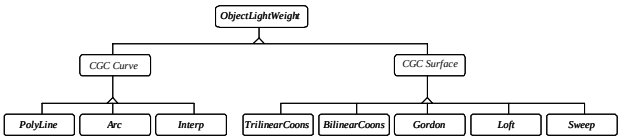

Figure 10: New classes implemented in the CGC representation.

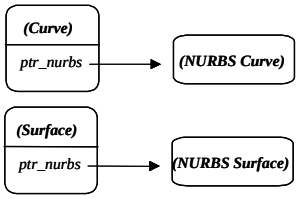

Figure 11: Relationship between the *Curve* and *Surface* classes of the CGC representation and the NURBS library.

6 Conclusions

This work described a modeling methodology focused on two aspects of the 3D finite element method simulation. The first aspect refers to non-manifold geometric modeling with automatic region detection (see, for example, Figures 12 and 13). In this context, a multi-region representation, called CGC representation, is generated by the insertion of curved surface patches. The second aspect is to provide support for automatic finite element mesh generation. This methodology is being implemented in an existing finite element modeler, called MG.

The paper also described an OOP class organization of a new version of MG that, while maintaining its simple and efficient user-interface, provides capabilities for automatic region detection. The OOP sketch was completed with the use of a NURBS library that models curves and surface patches (Figure 12).

To achieve this, a hybrid approach was adopted in which a full CGC representation of the model is not maintained in each step of the geometric modeling. Rather, the previous MG data structure was extended such that a CGC model can be created at any moment, when the user requires region detection or surface intersection.

Current developments are related to incorporating existing surface and solid finite element mesh generation algorithms to the system. The final goal is to develop a system for geometric modeling and automatic/adaptive 3D FEM simulation.

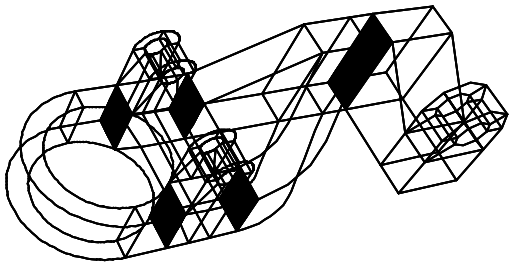

Figure 12: Modeling with multi-regions and NURBS.

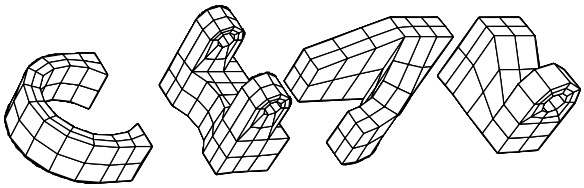

Figure 13: Geometric modeling – details of multi-regions.

7 Acknowledgements

The first author acknowledges a doctoral fellowship provided by CNPq. The authors acknowledge the finantial support provided by agency CNPq/CTPetro, process number 468447/2000-8, and by agency Finep/CTPetro, processes numbers 65999045400 and 650003600. The authors are grateful to Carolina Alfaro for the manuscript revision. The present work has been realized in Tecgraf/PUC-Rio (Computer Graphics Technology Group).

8 References

[1] O.C. Zienkiewicz and R.L. Taylor, *The Finite Element Method*, Fifth ed.. Vols. 1 and 2, Butterworth-Heinemann (2000).

[2] K.J. Bathe, *Finite Element Procedures*. Pretice Hill (1996).

[3] C.M. Hoffmann, *Geometric & Solid Modeling: An Introduction*. Morgan Kaufmann Publishers (1989).

[4] M.T.M. Carvalho, L.F. Martha and P.A. Wawrzynek, *An Architecture for Configuration of Geometric Modelers: Application to Computational Mechanics*. XXII SEMISH, Vol. 1, pp. 123-134, 1995 (in Portuguese).

[5] M.T.M. Carvalho, *A Strategy for the Development of Configurable Applications in Computational Mechanics.* PhD Thesis, Department of Civil Engineering, Pontifical Catholic University of Rio de Janeiro (1995) (in Portuguese).

[6] W.W.M. Lira, *A Configurable Integrated System for Simulations of Computational Mechanics Problems*. MSc Dissertation, Department of Civil Engineering, Pontifical Catholic University of Rio de Janeiro (1998) (in Portuguese).

[7] M. Mäntylä, *An Introduction to Solid Modeling Computer*. Science Press, Rockville (Maryland, 1988).

[8] L.C.G. Coelho, *Shell Modeling with Parametric Intersections*. PhD Thesis, Department of Informatics, Pontifical Catholic University of Rio de Janeiro (1998) (in Portuguese).

[9] A.C.O. Miranda, *Integration Finite Element Mesh Generation Algorithms*. MSc Dissertation, Department of Civil Engineering, Pontifical Catholic University of Rio de Janeiro (1999) (in Portuguese).

[10] J.B. Cavalcante Neto, P. A. Wawrzynek, M.T.M. Carvalho, L.F. Martha and A.R. Ingraffea, *An Algorithm for Three-dimensional Mesh Generation for Arbitrary Regions with Cracks*. *Engineering with Computers*, vol. 17, no. 1, pp. 75-91, 2001.

[11] L.C.G. Coelho and C.S. Souza, *Problems Communication and Geometric Solutions in a 3D Interface*. Proceedings of SIBGRAPI'95, pp. 183-190, 1995.

[12] L.C.G. Coelho, M. Gattass and L.H. Figueiredo, *Intersecting and Trimming Parametric Meshes on Finite-Element Shells*. International Journal for Numerical Methods in Engineering, vol. 47, no. 4, pp 777-800, 2000.

[13] P.R. Cavalcanti, P.C.P. Carvalho and L.F. Martha, *Non-manifold Modeling: An Approach Based on Spatial Subdivision. Computer-Aided Design*, vol. 29, no. 3, pp. 209-220, 1997.

[14] L. Piegl and W. Tiller, *The Nurbs Book*, 2nd ed. Springer (Ottawa, 1999).

[15] A.C.O. Miranda, J.B. Cavalcante Neto and L.F. Martha, *An Algorithm for Two-dimensional Mesh Generation for Arbitrary Regions with Cracks*. SIBGRAPI'99, pp. 29-38, 1999.

[16] A.G. Requicha, *Constructive Solid Geometry*. University of Rochester, Production Automation Project (1977).

[17] D.P. Dobkins and M.J. Laszlo, *Primitives for the Manipulation of Three-dimensional Subdvisions*. Third ACM Symposium on Computational Geometry, pp. 86-99 (Waterloo, 1987).

[18] M.J. Laszlo, *A Data Structure for Manipulating Three-dimensional Subdivisions*. PhD Thesis, Departament of Computer Science, Princeton University (1987).

[19] P. Lienhardt, *Extension of the Notion of Map Subdivisions of a Three-dimensional Space*. In STACS'88 Proceedings of the Cinquième Symposium sur les Aspects Thèoriques de L'Informatique, Bordeaux, 1988.

[20] K. Weiler, *Topological Structures for Geometric Modeling*. PhD Thesis, Rensselear Polytechnic Institiute, Troy (N.Y., 1986).

[21] K. Weiler, *The Radial-Edge Structure: A Topological Representation for Non-manifold Geometric Boundary Representations*. Geometric Modeling for CAD Applications, pp. 3-36 (North Holland, 1988).

[22] J.R. Rossignac and M.A. O'Connor, *A Dimensional-independent Model for Pointsets with Internal Structures and Incomplete Boundaries*. Geometric Modeling for Product Engineering, pp. 145-180, (North Holland, 1990).

[23] J.R. Rossignac and A.G. Requicha, *Constructive Non-regularized Geometry*. Computer Aided Design, vol. 23, no. 1, pp. 21-32, 1991.

[24] W.S. Ting, *Considerations about a Minimal Set of Non-manifold Operations*. Technical Memo, Technische Hochshule Darmstadt – GRIS, Wilhelminen-strabe, 7 DA-6100, FRG (1990).

[25] P. Lavoie, *NUSBS++: The Nurbs Package - User's Reference Manual – Version 3.0*, http://yukon.genie.uottawa.ca/~lavoie/software/nurbs/, Ottawa, 1999.

[26] P.R. Cavalcanti, *Creation and Management of Space Subdivisions*. PhD Thesis, Department of Computer Science, Pontifical Catholic University of Rio de Janeiro (1992).

[27] J. Rumbaugh, *Object-Oriented Modeling and Design*. Prentice-Hall (Englewood Cliffs, 1991).

[28] G.H. Paulino, I.F.M. Menezes, J.B. Cavalcante Neto and L.F. Martha, *A Methodology for Adaptive Finite Element Analysis: Towards an Integrated Computational Environment, Computational Mechanics*, vol. 23, no. 5/6, pp. 361-388, 1999.