ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN MATEMATICA
CICLO XXIX

Settore Concorsuale di afferenza: 01/A5

Settore Scientifico disciplinare: MAT/08

A NUMERICAL PROPOSAL OF AN EXTENDED SOLID MODELING SYSTEM

Presentata da: Giacomo Ferrari

Coordinatore Dottorato:
Chiar.ma Prof.ssa
Giovanna Citti

Relatore:
Chiar.ma Prof.ssa
Serena Morigi

Co-Relatori:
Chiar.mi Prof.
Giulio Casciola
Roberto Raffaeli

Esame finale 2017

A Margherita,
che mi ha sempre supportato
nei momenti più difficili

Abstract

This thesis has been carried out at the Department of Mathematics of the University of Bologna and at the Company HyperLean s.r.l. spin-off of the University Polytechnic of Marche, as part of a collaborative project on "Theoretical and numerical aspects of a hybrid geometric modeling system". The key observation that motivates the interest in this topic is that in different application contexts you have the need to create virtual solid models that integrate real data acquired by 3D scanning, represented by polygonal meshes, with synthetic models, designed by parametric/analytical multipatches.

The research topic covered the study of theoretical and numerical aspects of solid modeling and the development of suitable solutions as part of a "hybrid geometric solid modeling system".
In particular, the involvement as regards the professional side of the project covered the development of algorithms for the optimization of the 3D geometry of solid objects and boolean operations between polygonal meshes to improve the LeanCost software (HyperLean's proprietary software).
Concerning the academic side, we investigated many aspects of solid modeling, focusing on the B-Rep models and introducing the new paradigm "Extended B-Rep" which is able to integrate "mesh-faces" as part of a B-rep model.
To manage the quality of the built model we studied a notion of continuity and join between parametric and discrete representations and we proposed a set of methods that guarantee that the models can be manipu-

lated while maintaining predetermined continuity constraints among the constituent parts.

We generalized the most important tools of solid modeling to the Extended B-Reps and proposed solutions to extend the geometric kernels of standard solid modeling systems to be able to deal with Extended B-rep models. The new geometric solid modeling kernel has been realized in a software supported by the OpenCascade library.

Contents

Introduction

In different application fields, ranging from computer graphics to industrial design and 3D modeling, the user is faced with the design and editing of complex 3D virtual geometric models. Basically 3D models can be classified in two families: the digital models, that represent quite faithfully real objects or phenomena, and the designed models, which are a virtual representation of a shape concept, created by design. Each family has its own modeling pipeline. Digital models are the result of an acquisition process through 3D scanners. Oppositely, a designed model is typically created by means of a conventional computer aided modeling system, where a designer has at disposal a great number of tools to transpose his shape concept into a 3D model. Digital models are often represented by polygonal meshes or point clouds. Designed models are created by manipulating mathematical knowledge, such as Bézier and spline patches or analytical surfaces.
Up to now, only limited research efforts have been devoted to put together these two families of models and the only way that gives the possibility to these two categories to dial is the conversion of one into the other.
The conversion of different kinds of geometric primitives into a common form clearly implies expensive computations and possible loss of information. Given the complexity of the shapes to be virtualized, converting the digitized meshes into continuous designed models is unfeasible. On the other hand, if one chooses to convert continuous surfaces into polygonal meshes through tessellation, editing the model afterwards, if necessary, will be harduous. Moreover, the digital preservation of acquired pieces requires that one can

always distinguish digital models from the surface patch created following a geometric process.

This project aims at integrating designed models, represented by continuous surfaces, and digital models, represented by polygonal meshes, in a unique 3D model in which entities such as Non Uniform Rational B-Splines (NURBS) analytical surfaces and meshes coexist. This would delete the gap existing between designed and digital models and it would simplify many processes that nowadays require the conversion of one representation in the other.
To support this new modeling paradigm we propose a suitable solid modeling system that we name "Extended Solid Modeling System". It is pointing out that the proposed term "Extended" instead of Hybrid avoids a possible misunderstanding. Infact, nowadays, what is commonly referred to as hybrid system is a modeling system in which it is possible to model together solid objects and surfaces. The adjective Extended is in order to underline the extension which was made compared to existing systems.
In the proposed Extended Solid Modeling System, which relies on Boundary Representation (B-Rep) of solid models, the faces are described by different kinds of representations, both continuous and discrete, such as NURBS surfaces and meshes. Here different forms of representation coexist, interact and, since they do not have to be converted into a common form, they always keep their shape features and analytical properties. The regions of the model represented by meshes maintain their faithful compliance to the real data, while those represented by continuous models are easily editable.
The possible applications of such an Extended System are spread through cultural heritage, medical science, passing through industrial design and engineering applications. Every scenario in which it is necessary to build or rebuild a piece or a surface starting from an existing model represented by a mesh is a possible application for our proposal.

In the context of cultural heritage, artwork findings are not only fragmented, but are often incomplete. Therefore, to build a virtual model of a restored object, the digitized data must be supplemented by modeling the missing parts and assembling them in such a way to be compliant with the real findings. Currently available systems address the problem by converting different representations into a common form. Unfortunately, in the virtualization of complex artworks, the conversion of detailed meshes into smooth surfaces implies unacceptable approximations, while the conversion of smooth surfaces into meshes makes it difficult to operate on the model to make changes.
Instead, an extended model would allow to combine the expressiveness of meshes with the easy editability of smooth surfaces in order to restore the missing parts. No conversion would be required.
An example is illustrated in Fig.1 and Fig.2 where are represented two dif-

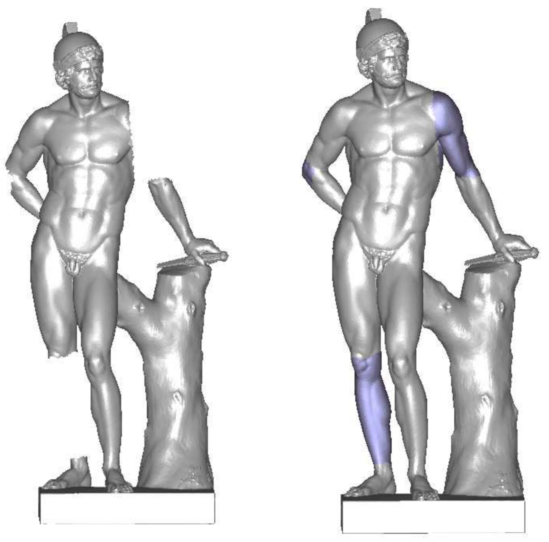


Figure 1: Example of Statue reconstruction, Mars statue is repaired

ferent ruined statues. In Fig.1(a) is illustrated the statue ruined with some missing pieces, while in Fig.1(b) the statue has been repaired recreating miss-

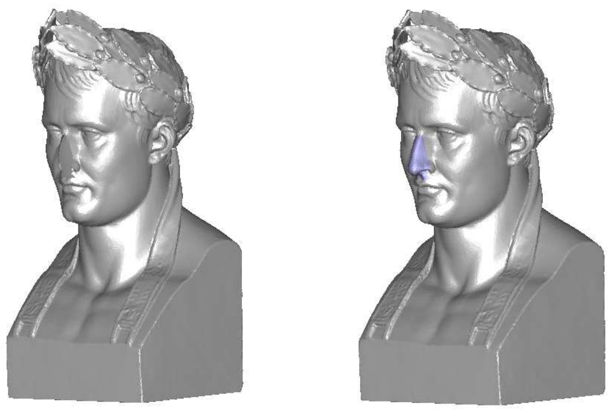


Figure 2: Example of statue reconstruction, Caesar's Nose recreated and attached

ing parts using NURBS surfaces.
Another example is shown in Fig.2 which shows how a nose is reconstructed using a CAD system and then joined to the rest of the statue. In Fig.1 and Fig.2 from left to right, we can see respectively the ruined statue and the repaired one.

A different application that motivated the research proposed is in the context of biomedical engineering. In particular the implant design, Plastic Surgery and Maxillofacial Surgery. The extended paradigm could be used to integrate the digitalized parts of the human patient with parts modeled by a biomedical designer. This would make the design and prototyping processes quicker, cheaper and more efficient.
By the way of illustration we show an example in Fig.3(a) of broken femur. Fig.3(b) illustrates a possible prosthesis that can be modeled with a CAD system. In this case an extended model could be used to adapt exactly the boundary of the sphere to the boundary of the broken femur in order to have an accurate result.

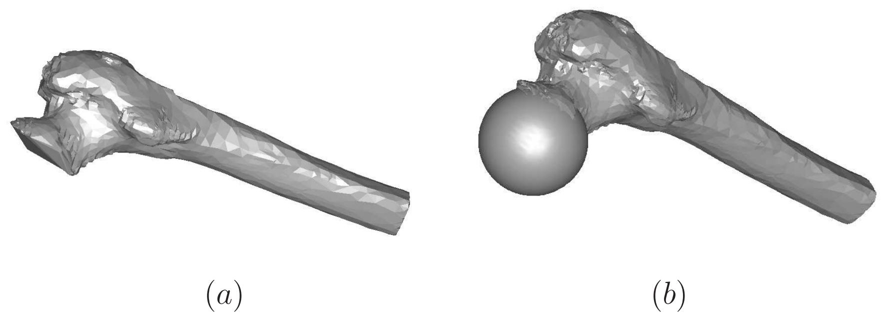


Figure 3: Example of femur reparation

To formulate a complete proposal of an Extended Solid Modeling System we investigated theoretical and numerical aspects and we developed suitable software tools. This project has involved two main goals:

a) Study and design of a theory for a new paradigm of extended solid model reprentation. This will provide the basis for the definition of a new paradigm of an Extended Solid Modeling kernel.

b) Development of tools to extend a classic solid modeling system, aimed at integrating the new primitives and the new paradigm.

a) Since digital models are represented by meshes, while designed models exploits continuous surfaces, the main proposal is the study of a new reprentation scheme, named EB-Rep, that extends the classic B-Rep, in which meshes are represented using a new entity called Mesh-Face.
Moreover we study the Boolean Operation between Extended models, investigating the mesh/NURBS intersection algorithm, in order to realize new mixed models in which NURBS and meshes coexist.
Then, to control the quality of the model, it is necessary to define a notion of continuity for Extended models, that we call approximate geometric continuity (AG), and a set of conditions to guarantee that extended models be manipulated keeping prescribed continuity constraints between their constituting patches. To this aim we investigate how to handle the join between

mesh and NURBS entities.
b) In particular we tackled the definition of a new data structure for extended models and develop algorithms for the join between mesh and NURBS.
Beyond the design of such a new Extended Solid Modeling System, we investigated the instruments to integrate our innovative proposal in actual systems, giving the possibility to enlarge their potentialities without strongly modify their geometric kernel. Although the activities of the project have been mostly oriented to a fundamental research, some efforts have been also addressed to the development of a prototype software, which is an extension of a classic solid modeling system that manages both NURBS surfaces and meshes. This system will be used to validate the effectiveness of the proposed theory and methodologies.

The work of this thesis is organized as follows.
In the first chapter we introduce the basic notions necessary for our work. In particular we briefly discuss the representation schemes, focusing on B-Rep representations. Then we introduce parametric curves and surfaces and, in particular, NURBS and finally meshes and their properties.
In the second chapter we formalize the innovative concepts of Mesh-Face and Extended B-Rep model (EB-Rep).
In the third chapter we describe methods to efficiently represent valence semi-regular meshes with NURBS surfaces in an EB-Rep model.
In the fourth chapter we introduce methods to efficiently represent an unstructured mesh with NURBS surfaces in an EB-Rep model.
In the fifth chapter we introduce the most important tools of solid modeling and show how to adapt them for an Extended B-Rep model. In particular we consider Boolean Operations, Cutting Operation and Face-Join for Extended B-Rep.
In the sixth chapter we illustrate in details an example of Finite Element Analysis applied to an Extended B-Rep solid. In particular we apply the Finite Cell Method to an Extended B-Rep solid obtained with tools of solid

modeling introduced in the fifth chapter.
In the last chapter we illustrate in details the research activity performed at Hyperlean company. In particular we describe the tools realized in order to improve LeanCost software. These tools are a first application of innovative concepts introduced in our Extended Solid modeling System.

Chapter 1

Geometric Representation of Solid Objects

A Solid Model is a digital representation of the geometry of an existing or envisioned physical object. Solid models are used in many industries ranging from manufacturing to health care. Solid Modeling is a consistent set of paradigms and algorithms for the representation and construction of solid objects.

Principles of geometric and solid modeling are the foundation of Computer Aided Design (CAD) and in general support the creation, exchange, visualization and interrogation of digital models of physical objects. In particular solid modeling techniques allow for the automation of several difficult engineering calculations that are an important part of the design process. Simulation, planning and verification of machining and assembly processes were one of the main reasons for the development of solid modeling. In addition, solid modeling techniques serve are the basis for rapid prototyping, reverse engineering and mechanical analysis using finite elements.

A central problem in all these applications is the ability to effectively and unambiguously represent and manipulate three-dimensional geometry in order to have a representation consistent with the physical behavior of real objects. Solid modeling research and development has effectively addressed

many of these issues, and continues to be a central focus of Computer-Aided Engineering (CAE). The Solid Modeling technology is implemented in several commercial solid modeling software systems.
Solid modeling is an interdisciplinary field that involves many areas, from rigorous mathematical theories, computational geometry to expects of computer aided geometric designer. Moreover the computational aspects of solid modeling deal with efficient data structures and algorithms.

In this chapter we introduce some basic notions on the mathematical representation of solid objects. In particular we introduce the definitions of Representation Schemes, focusing on B-Rep representations, geometric notions of parametric curves and surfaces and, in particular, NURBS. In the last part another important geometric primitive, the polygonal mesh, is introduced.

1.1 Representation Scheme

Solid modeling relies on the specific need for informational completeness in mechanical geometric modeling systems, in the sense that any digital model should support all geometric queries that may be required about its corresponding physical object. This necessity led to the development of the modeling paradigm that has defined the field of solid modeling as we know it today [45]. These paradigms are based on the Representation Schemes. Let's introduce some basic definitions.

Definition 1.1. *A* R-set *is a subset of 3D Euclidean space that is closed, bounded, regular and semianalityc.*

A physical object, modeled mathematically by an R-set, is unambiguously defined by its boundary. All the R-set's properties allow to study these entities as if they were real solids.

Definition 1.2. *A* Syntactically Correct Representation *is a finite symbol* *structure constructed with symbols from an alphabet according to syntactical rules.*

The collection of all syntactically correct representations is called a *Representation Space* **R**. The mathematical modeling space whose elements are R-sets (abstract solids) is called **N**.

Definition 1.3. *A representation scheme is a relation:*

$$rs : \mathbf{N} \longrightarrow \mathbf{R}$$

Let **D** be the set of the representable solids in rs and **V** be the set of valid representations in **R**.
All the principal solid representations can be associated with rs and can have the following properties:

. Domain: the set **D** of representable solids in rs. If the representation is optimal we have $\mathbf{D} = \mathbf{N}$

. Validity: impossibility to create a non-sense representation

. Completeness: all representations are not-ambiguous. This means rs^{-1} is a function

. Uniqueness: every solid has a unique representation. This means rs is a function

. Accuracy: it's possible to represent exactly a solid

. Simplicity: it's easy to create a representation

. Efficiency: closure, robust algorithms, compact storage.

The principal representation schemes used in CAD and CAGD application fields are Constructive Solid Geometry (CSG) and Boundary Representations (B-Rep). Moreover there are other representation schemes used for different applications which are, for example:

- Voxelization: it is essentially a list of spatial cells occupied by the solid.
- Cell decomposition: the solid is represented by its decomposition into several cells.
- Surface Mesh modeling: the boundary of the solid is discretized using a mesh.
- Sweeping: the solid is defined by a set moving through space that trace or sweep out volume.
- Implicit representation: the solid is specified by a predicate (in/out) that can be evaluated at any point in the space.

The most important CAD systems and solid modeling libraries use CSG and B-Rep representation schemes. In particular, the open source softwares Blender [2], OpenCascade [5] and CGAL [3] use B-Rep data structure.

Constructive Solid Geometry CSG is a representation scheme in which a solid is described through basic primitives, combined using boolean operators and rigid motions as shown in the example in Fig.1.1. Often a CSG model appears visually complex, but is actually a clever combination or decombination of simple objects called primitives.
Primitives are the simplest solid objects used for the representation. Typically they are simple shape like cuboids, cylinders, prisms, pyramids, spheres and cones.
This work focus on Boundary Representation schemes which are discussed more in details in the next section.

1.2 Boundary Representation (B-Rep)

A B-Rep representation describes a solid using the decomposition of its boundary in a collection of connected surface elements. The boundary of the solid separates the inner from the outer space. Every point in space can

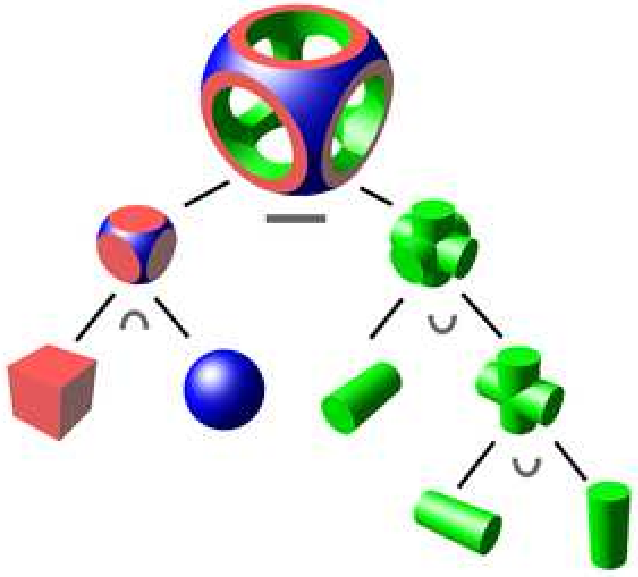


Figure 1.1: CSG Representation Scheme: Union ($\cup$), Intersection ($\cap$) and Difference ($-$) boolean operations used to construct a solid object from simple primitives (sphere, cone, cube)

unambiguously be tested with respect to the solid by testing the point with respect to the boundary of the solid. This allows us to test the validity of the representation. The boundary is described by a pair of sets: the set of geometric entities and the set of topological information. Geometric entities are surfaces, curves and vertices. Topological entities are faces, edges and vertices that are associated with the geometric entities. Connections between topological entities give a detailed description of the shape of the object. More in detail a B-Rep represents a solid describing the relationship between geometric entities.

Definition 1.4. *A* B-Rep scheme $B = (G, T)$ *consists of*

. *a set of geometric data* $G = (P, C, S)$*, where* P *contains points in* $\mathbb{R}^3$*,* C *contains parametric curves in* $\mathbb{R}^3$ *and* S *contains analytic and parametric surfaces* $\mathbb{R}^3$*.*

. *a set* $T = (V, E, F)$ *of topological information providing relationships* *among the elements of G, where V are the vertices, E are the edges and F are the faces.*

The entities in G are:

. Points (P): it contains points P_i in $\mathbb{R}^3$ that are associated with vertices in V.

. Curves (C): it contains curves C_i in $\mathbb{R}^3$ with their parameterization $p(t) \in C_i, t \in [t_1, t_2]$. A curve is associated with an edge in E.

. Surfaces (S): it contains parametric surfaces S_i in $\mathbb{R}^3$ with their parameterization $p(u, v) \in S_i$. A surface is associated with a face in F.

The entities in T are:

. Vertices (V): it contains vertices V_i that are pointers to the associated point P_i in $\mathbb{R}^3$ defined in G.

. Edges (E): it contains E_i that are pointers to the associated curves C_i defined in G.

. Faces (F): it contains F_i that are pointers to the associated surfaces S_i defined in G.

. Loops/Wires (W): it is an ordered sequence of vertices and edges. A loop defines a not self-intersecting piecewise closed space curve W which may be a boundary of a face. A loop can be considered as a particular closed edge.

. Bodies/Shells (B): it is a set of faces that bound a single connected closed volume. It's possible to define a Skeletal Body as a solid made of a unique point. This solid has a face with no boundary.

As shown in the simple example in Fig.1.2, all information about the cube are stored in a Hierarchical Linked Table, shown in Fig.1.2(b), in which geometric information (points, curves and surfaces) are linked by topological entities (vertices, edges and faces). The faces $A, B, \ldots, F$ are delimited by the edges $l1, \ldots, l12$. For example, according to the linked table, face A is delimited by edges $l1, l2, l3, l4$. The edge $l1$ is delimited by vertices $v1$ and $v2$.

A more complex example is illustrated in Fig.1.3(c). The topological struc-

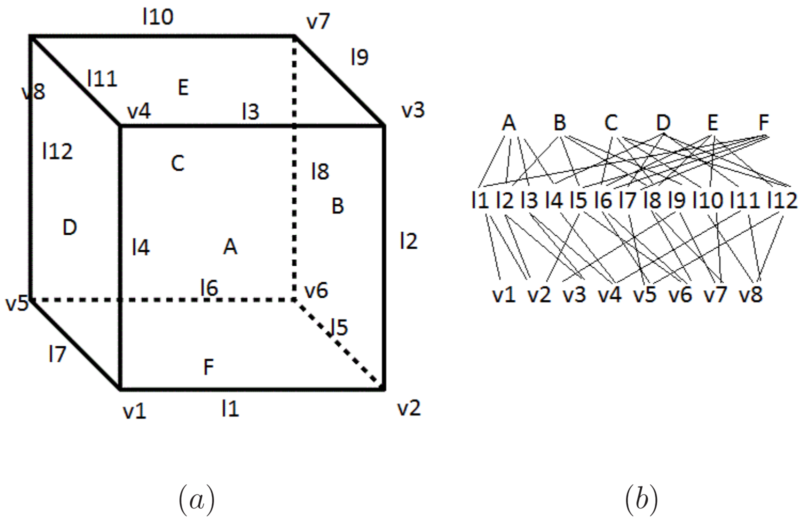


Figure 1.2: Example of B-Rep Scheme: a) a solid object b) hiearchical linked table associated with the object

ture of the object is represented in Fig.1.3(a). Fig.1.3(b) shows the geometric entities associated with every face topological entity which are NURBS surfaces that bound the solid object. Most B-Rep schemes store additional information to accelerate the traversal and processing of the boundary.

There are three different B-Rep classes: vertex-based-B-Rep, edge-based-B-Rep and face-based-B-Rep.

A *vertex-based* representation is the simplest B-Rep. In this scheme faces are stored using a counterclockwise ordered list of vertex. Geometric information about vertices are stored in a linked table.

An *edge-based* representation has the edge as fundamental geometric entity. Edge-base data structures are the B-Rep schemes that allow to store the

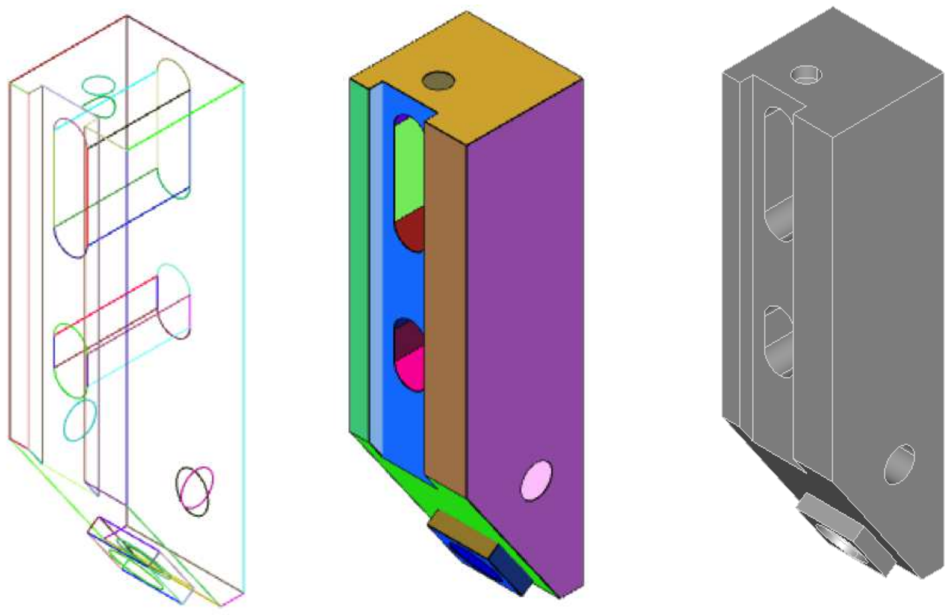


Figure 1.3: Example of B-Rep Representation Scheme: a) Topological structure, b) Geometric entities, c) Model

maximum amount of information about the represented solid.
In a *face-based* representation a graph is used to represent the connections between the faces of a solid object.
In the next two subsections we focus on two particular edge-based data structures used to handle topology of the B-Rep schemes: Winged-Edge and Half-Edge. These are the most frequently used and the most efficient due to their compactness capabilites to store all necessary pieces of information in a small number of elements.

1.2.1 Winged-Edge data structure

A winged-edge data structure represents a particular edge-based B-Rep in which every edge stores four data:

. Pointers to its two end vertices

. Pointers to left and right faces (in manifold solids)

. Pointers to preceding and next edge in clockwise order

. Pointers to preceding and next edge in counterclockwise order

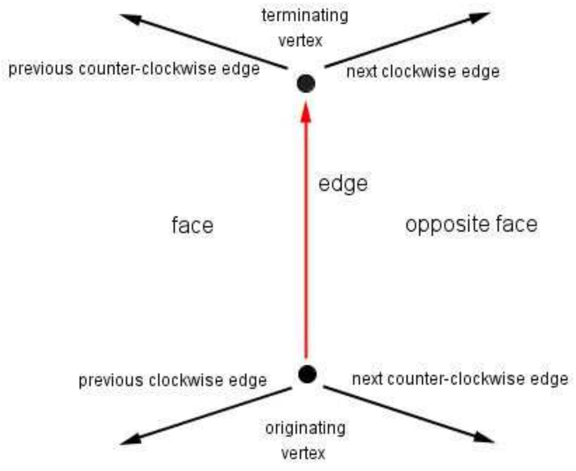


Figure 1.4: Winged-Edge data structure

An example of winged-edge data structure is illustrated in Fig.1.4. Each vertex and each face has a pointer to one reference edge.
Each edge is oriented: its orientation is given by its originating and terminating vertices. Clockwise and counterclockwise orientations are given with respect to the orientation of the edge.
The winged-edge data structure has been designed to allow effective local modification of the solid topology. It is the oldest B-Rep data structure and was initially used for representing polygonal meshes. The basic winged-edge data structure assumes that every edge of the model has exactly two adjacent faces. This restricts the topology of surfaces to be 2-manifolds without boundaries. A manifold model only contains manifold surfaces. However, the result of boolean operations on manifold solids can lead to a result that

is manifold with boundaries. An advanced winged-edge data structure allows to represent also solid manifolds with boundary and non-manifolds.

1.2.2 Half-Edge data structure

Modern solid modelers use an advanced edge-based B-Rep data structure: the half-edge data structure. In this representation scheme an edge is divided in two coincident half-edges with the same shape, the same ending vertices with opposite orientation. Half-edge is the main geometric entity of the structure and stores five information:

. A pointer to the previous half-edge

. A pointer to the starting/ending vertex

. A pointer to the incident face

. A pointer to the opposite half-edge

As illustrated in Fig.1.5 each vertex and each face store a pointer to the associated half-edge. The half-edge associated with a face is considered the first edge of the face's border loop. This scheme allows to represent faces orientation.

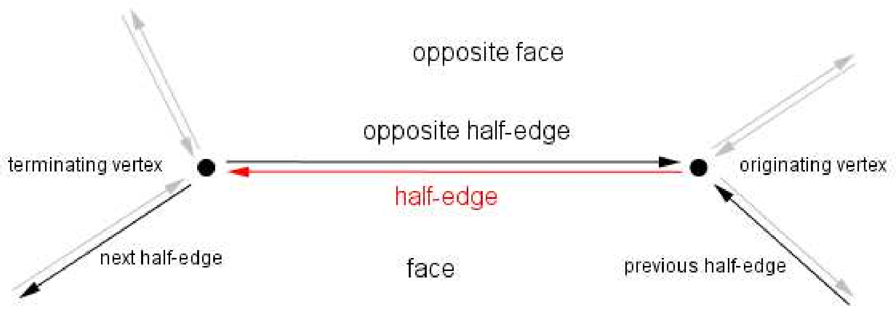


Figure 1.5: Half Edge data structure

1.3 Euler's Equation

The Euler's Equation describes the relationship between geometric entities and topological properties of the represented solid. More in detail, a representation of a 2-manifold without boundary is considered a valid representation if it satisfies the following relation:

$$|F| - |E| + |V| - |L| = 2(|C| - G) = \chi \tag{1.1}$$

where

- $|F|$ is the number of Solid's Faces
- $|E|$ is the number of Solid's Edges
- $|V|$ is the number of Solid's Vertices
- $|C|$ is the number of Solid's Connected Bodies
- $|L|$ is the number of Inner Loops in every face
- G is the genus of the Solid

The Euler Characteristic (or Euler Number), denoted by χ, is a topological invariant, a number that describes a topological shape in the space or a structure regardless of the way it is bent. In case of connected solids it provides a direct link to the topological genus of the object. Eq.(1.1) is not the classical Euler's Equation because it includes also the number C of connected components, assuming that the solid can have more than a single connected component. We refer to [43] for details on this equation and on a more general Euler's Equation that describe Non-Manifold Solids and Solids Manifold with Boundary.

As a simple example, we apply Euler's Equation to the solid cube illustrated in Fig.1.6, where the cube is described by 6 quadrilateral faces (Fig.1.6(a)) and by 12 triangular faces (Fig.1.6(b). It is a solid with a single connected component and genus 0, and we have:

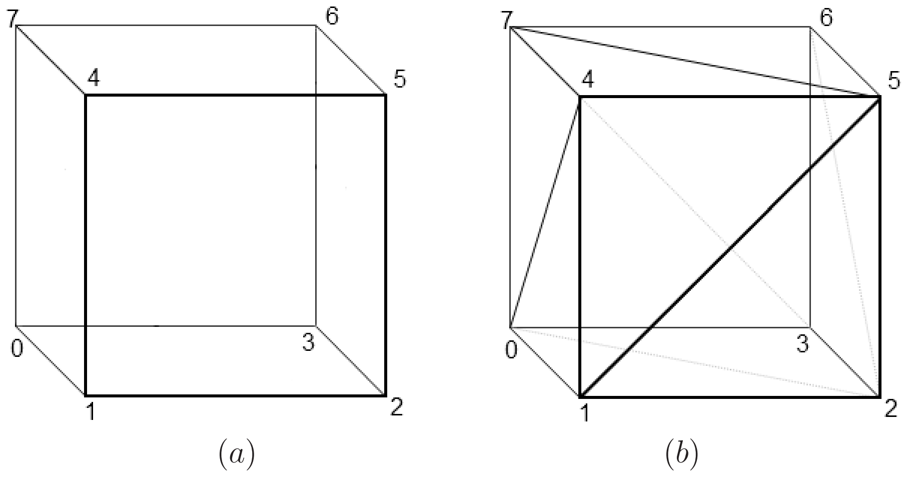


Figure 1.6: Cube represented with 6 quad-faces (a) and 12 triangular faces (b)

. 6 Faces, 12 Edges, 8 Vertices in the quadrilateral case

. 12 Faces, 18 Edges, 8 Vertices in the triangular case

According to Euler's Equation (1.1) we have :

$$6 - 12 + 8 = 2(1 - 0) = 2$$

for the quad-case and

$$12 - 18 + 8 = 2(1 - 0) = 2$$

for the triangular one. Both representations result in $\chi = 2$ that is both are *valid* representations of a cube.

Each face of a B-Rep can be associated with a geometry which characterizes its shape. In particular, standard B-Rep systems use analytical surfaces such as planes, spheres, cylinders and spline parametric surfaces.
In the following we briefly describe the spline parametric surfaces and the

meshes, that are introduced as a new geometric primitive in the proposed Extended B-Rep system and they define faces of the new proposed B-Rep extension scheme.

1.4 Parametric curves and surfaces

Parametric curves and surfaces are the most used geometric entities in a classical B-Rep scheme and are associated to edges and faces respectively.

Definition 1.5. *Let I be an interval of $\mathbb{R}$. A* curve *γ is a continuous mapping $\gamma : I \to X$, where X is a topological space.*

. *γ is said to be simple, or a Jordan arc, if it is injective, i.e. if for all x, y in I, we have $\gamma(x) = \gamma(y)$ implies $x = y$*

. *If $\exists x, y$ with $x \neq y$ such that $\gamma(x) = \gamma(y)$ (with x,y different from the extremities of I), then $\gamma(x)$ is called a double (or multiple) point of the curve.*

. *A curve γ is said to be closed or a loop if $I = [a, b]$ and if $\gamma(a) = \gamma(b)$.*

We always consider curves in $n-$dimensional Euclidean spaces, with $n \geq 1$. In particular we consider plane curves (on $\mathbb{R}^2$) and space curves (on $\mathbb{R}^3$).

Definition 1.6. *A* parametric curve *$c(t)$ is a geometric entity whose equations express the coordinates of points as function of a variable t, called parameter:*

$$c : \mathbb{R} \to \mathbb{R}^n$$

where $n = 2, 3$.

According to this definition, we write $c(t) = (x(t), y(t))$ if $n = 2$ or $c(t) = (x(t), y(t), z(t))$ if $n = 3$, where x, y, z are the three components of the function and represent the coordinates of the point in $\mathbb{R}^2$ or $\mathbb{R}^3$ associated with the parameter t.

In order to define continuity of curves we need to introduce differentiability of functions. Let's now consider k a non negative integer and D an open set on the real line such that f is defined on that set with real values.

Definition 1.7. *A function f is said to be of* (differentiability) class C^k *if the derivatives f', f'', ..., $f^{(k)}$ exist and are continuous (the continuity is implied by differentiability for all the derivatives except for $f^{(k)}$).*
f is said to be of class C^∞, or smooth, if it has derivatives of all orders.

Continuity of parametric curves is described by parametric continuity as follow.

Definition 1.8. *A curve can be said to have C^n* continuity *if*

$$\frac{d^n\gamma}{dt^n}$$

is continuous of value throughout the curve.
This means that the first n derivatives of the functions that describe the curve are continuous.

According to the definition, a curve is said to be C^0 if the curve is continuous, C^1 if first derivatives are continuous, C^2 if first and second derivatives are continuous and C^n if first, second, ..., $n-$th derivatives are continuous. In general, when two joining curves describe a 2D/3D shape, the requirement of C^n continuity at the contact point can be quite restrictive.
In addition to parametric continuity, geometric continuity (G^n) was introduced to make the shape description independent on the speed to trace out the curve. In particular geometric continuity describes continuity between two parametrizations q and r of two curves joined at an extreme point considering also the equivalent parametrizations of q and r [29].

Definition 1.9. *Let $q(u)$, $u \in [a, b]$, and $\tilde{q}(\tilde{u})$, $\tilde{u} \in [\tilde{a}, \tilde{b}]$, be two regular C^n parametrizations. These parametrizations are said to be* equivalent, *that is, they describe the same oriented curve, if there is a C^n function $f : [\tilde{a}, \tilde{b}] \to [a, b]$ such that:*

. $\tilde{q}(\tilde{u}) = q(f(u))$

. $f(\tilde{a}) = a \quad f(\tilde{b}) = b$

. $f' > 0$

Definition 1.10. *Let's consider q and r be two C^n parametric curves meeting at a point P. They meet with $n-$*th-order geometric continuity*, denoted by G^n, if there exists a parameterization $\tilde{q}$ equivalent to q such that $\tilde{q}$ and r meet with C^n continuity at P.*

An example is shown in Fig.1.7. On the left two curves join with a G^1 connection: the tangent vectors have the same direction but different length. On the right they join with C^1 connection: the tangent vectors have the same length and direction.
According to the definition, if we consider the tangent vectors on both sides

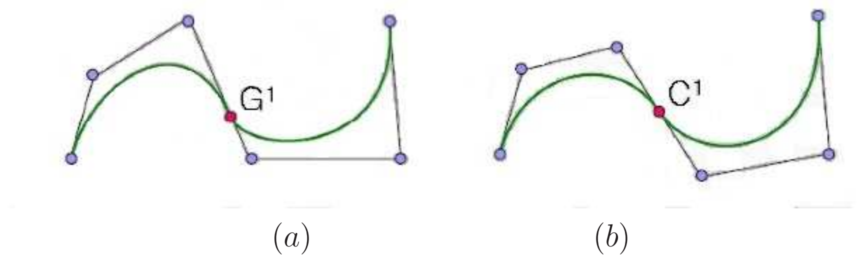


Figure 1.7: Join between two curves represented in Bernstein Basis form with continuity G^1 (a) and C^1 (b)

of a point on a curve, they are G^0/C^0 connected if the curves touch at the join point, G^1 connected if the curves share a common tangent direction at the join point, G^2 connected if the curves also share a common center of curvature at the join point and G^n connected if $q^{(n)}(t) \neq 0$ and $q^{(n)}(t) = kg^{(n)}(t)$, for a scalar $k > 0$.
In general, G^n continuity exists if the curves can be reparametrized to have C^n (Parametric) continuity. A reparametrization of the curve is geometrically identical to the original, only the parameter is affected.

We now introduce basic definitions for parametric surfaces.

Definition 1.11. *A* surface *s is a topological space in which every point has an open neighbourhood homeomorphic to some open subset of the Euclidean plane E^2.*

Definition 1.12. *A* parametric surface *is the image of an open subset of the Euclidean plane (typically $\mathbb{R}^2$) by a continuous function in a topological space that is generally an Euclidean space of dimension at least three.*

In our work we consider surfaces $s : \mathbb{R}^2 \to \mathbb{R}^3$ defined by vector functions of two variables u and v, called parameters, as follows:

$$s(u, v) = (x(u, v), y(u, v), z(u, v)),$$

where x,y,z are the three components bivariate functions that represent the coordinates of the point in $\mathbb{R}^3$ with parametric coordinates (u, v).
The Jacobian matrix is the matrix of all first-order partial derivatives. Jacobian matrix of a parametric surface is a 3×2 matrix. A point p whose Jacobian matrix has rank two is said regular, or the parametrization is said regular at p. A surface has C^1 continuity if the Jacobian matrix associated to the surface has rank 2 for all points throughout the surface itself.
The tangent plane at a regular point p is the unique plane passing through p, and is generated by the two row vectors of the Jacobian matrix. The normal line, or simply normal at a point of a surface is the unique line passing through p and perpendicular to the tangent plane. A normal vector is a vector which is parallel to the normal. According to these notions, surfaces are said to be:

. C^0 if the result surface is continuous

. C^1 if the tangent plane exists for every internal point of the result surface.

According to [29] we give the following definition.

Definition 1.13. *Let s and r be two C^n parametric surfaces meeting along an edge e. They meet with n-*th-order geometric continuity*, denoted G^n, if there exists an equivalent reparametrization $\tilde{s}$ of s such that, along e, $\tilde{s}$ and r meet with C^n continuity.*

1.5 NURBS Curves and Surfaces

Non Uniform Rational B-Spline (NURBS) are mathematical models used in geometric modeling to generate and represent parametric curves and surfaces. Their properties allow to have a great flexibility and precision for handling both analytic and free-form shapes. NURBS are commonly used in CAD, CAM and Computer Aided Engineering (CAE) and they are represented by the common standards, such as IGES, STEP and others.
In order to define NURBS curves and surfaces we need the following definitions. For properties and tools we referred to [28] or [50].

Definition 1.14. *Let $[a, b]$ be a closed and bounded interval and $\Delta = \{x_i\}_{i=1,\dots,k}$ a set of points (knots) such that:*

$$a \equiv x_0 < x_1 < \dots < x_k < x_{k-1} \equiv b$$

We denote by Δ the partition of $[a, b]$ in $k + 1$ subintervals:

- $I_i = [x_i, x_{i+1}) \qquad i = 0, \dots, k-1$
- $I_k = [x_k, x_k + 1]$

Given an integer $m > 0$, and $\mathbb{P}_m$ the space of real polynomials of order at most m, we define the space of piecewise polynomials*:*

$$\mathbb{PP}_m(\Delta) = \{f | \exists p_0, \dots, p_k \in \mathbb{P}_m \ s.t \ f(x) = p_i(x) \quad \forall x \in I_i, \quad i = 0, \dots, k\}$$

Definition 1.15. *Let $[a, b]$ be a closed and bounded interval and $\Delta = \{x_i\}_{i=1,\dots,k}$ a partition of $[a, b]$. Let m be a positive integer, $M = (m_1, m_2, \dots, m_k)$*

a vector of positive integers such that $1 \leq m_i \leq m, \quad \forall i = 1, \ldots, k$. *Let* $W = (w_1, \ldots, w_k)$ *be a vector of not negative coefficients. We define space of polynomial B-Spline functions of order* m *with knots* $x_1, \ldots, x_k$ *of multiplicity* $m_1, \ldots, m_k$ *as follows:*

$$S(\mathbb{P}_m, M, \Delta, W)) \quad = \quad \{s(x) \mid \exists s_0(x), \ldots, s_k(x) \in \mathbb{P}_m \ s.t.$$

$$s(x) = s_i(x) \quad x \in I_i \quad i = 0, \ldots, k$$
$$s^l_{i-1}(x_i) = s^l_i(x_i) \quad l = 0, \ldots, m - m_i - 1 \quad i = 1, \ldots, k\}.$$

We observe that for $m_i = 1 \ \forall \ i = 1, \ldots, k$ we have the maximum continuity, instead if $m_i = m \ \forall \ i = 1, \ldots, k$, $S(\mathbb{P}_m, M, \Delta, W)$ reduces to $\mathbb{PP}_m(\Delta)$. The space $S(\mathbb{P}_m, M, \Delta, W)$ has dimension $m+K$, where $K = \sum_{i=1}^k m_i$. In the same way we can define the space of rational B-Spline functions as follows.

Definition 1.16. *The* space of rational B-Spline functions *is represented as*

$$R(\mathbb{P}_m, M, \Delta, W)$$

where $M = (m_1, m_2, \ldots, m_k)$ *is the multiplicity vector,* $W = (w_1, \ldots, w_k)$ *the vector of weights and* $m + K$, *with* $K = \sum_{i=1}^k m_i$, *is the dimension of the space.*

Definition 1.17. *The set*

$$\Delta^* = \{t_i\}_{i=1,\ldots,2m+K}$$

where $K = \sum_{i=1}^k m_i$ *is called* Extended Partition associated with $R(\mathbb{P}_m, M, \Delta, W)$ *if and only if:*

. $t_1 \leq t_2 \leq \ldots \leq t_{2m+K}$

. $t_m \equiv a$; $t_{m+K+1} \equiv b$

. $t_{m+1}, t_{m+2}, \ldots, t_{m+K} \equiv (x_1, \ldots, x_1, \ldots, x_k, \ldots, x_k)$ *where* x_i *is repeated* m_i *times* $\forall i = 1, \ldots, k$

A stable base for the space $S(\mathbb{P}_m, M, \Delta, W)$ is represented by the Normalized B-Spline functions.

Definition 1.18. *Let Δ^* be the extended partition associated with $[a, b]$. We define the set of* Normalized B-Spline functions

$$\{N_{i,m}\}_{i=1,\dots,m+K}$$

using the recursive formula:

$$N_{i,h}(x) = \begin{cases} \dfrac{x - t_i}{t_{i+h-1} - t_i} N_{i,h-1}(x) + \dfrac{t_{i+h} - x}{t_{i+h} - t_{i+1}} N_{i+1,h-1}(x) & \text{\textit{if} } t_i \leq t_i + h \\ 0 & \text{\textit{otherwise}} \end{cases}$$

for $h = 2, \dots, m$, where:

$$N_{i,1}(x) = \begin{cases} 1 & if\ t_i \leq x\ \leq t_{i+1} \\ 0 & otherwise \end{cases}$$

and conventionally $\frac{0}{0} = 0$.

Definition 1.19. *Given $S(\mathbb{P}_m, M, \Delta, W)$ of dimension $m + K$, we define a* B-Spline function *of order m in $S(\mathbb{P}_m, M, \Delta, W)$ as:*

$$f(x) = \sum_{i=1}^{m+K} c_i N_{i,m}(x), \tag{1.2}$$

where $x \in [a, b]$ and $\{N_{i,m}(x)\}_{i=1}^{m+K}$ are the Normalized B-Spline basis functions defined on Δ^*.

Definition 1.20. *Given $R(\mathbb{P}_m, M, \Delta, W)$ of dimension $m + K$, we define a* Rational B-Spline function *in $R(\mathbb{P}_m, M, \Delta, W)$ as:*

$$r(x) = \frac{\displaystyle\sum_{i=1}^{m+K} c_i w_i N_{i,m}(x)}{\displaystyle\sum_{i=1}^{m+K} w_i N_{i,m}(x)}, \tag{1.3}$$

where $x \in [a,b]$, $\{N_{i,m}(x)\}_{i=1}^{m+K}$ are the Normalized B-Spline basis functions defined on Δ^* and $w_i > 0\ \forall i$.

In order to modify a B-Spline function and enlarge the dimension of its space, we can act on the knot vector or on the degree. In these case coefficients c_i of the function are modified but the function shape is unchanged. We introduce the most important tools used to modify the knot vector of a B-Spline function. These methods are valid also for Rational B-Spline functions.

Knot Insertion The Knot Insertion (KI) algorithm modifies the knot vector of a B-Spline function $f(x)$ adding one or more knots to the extended partition Δ^* associated with $f(x)$. Knot Insertion leaves $f(x)$ unchanged.
Given Δ^* the extended partition associated with $[a,b]$ and $\hat{t} \in [a,b]$, we insert a knot $\hat{t}$ in Δ^*, where $t_l < \hat{t} < t_{l+1}$, thus generating a new extended partition $\hat{\Delta}^* = \{\hat{t}_i\}_{i=2m+K+1}$ where:

$$\hat{t}_i = \begin{cases} t_i & i \leq l \\ \hat{t} & i = l+1 \\ t_{i-1} & i \geq l+2 \end{cases}$$

The new space of functions $S(\mathbb{P}_m, \hat{M}, \hat{\Delta}, \hat{W})$ has dimension $m+K+1$ and is such that $S(\mathbb{P}_m, M, \Delta, W) \subset S(\mathbb{P}_m, \hat{M}, \hat{\Delta}, \hat{W})$.
The Normalized B-Spline basis functions are modified according to the following theorem introduced in [15].

Theorem 1.5.1. *Given Δ^* and $\hat{\Delta}^*$ the two extended partitions defined before, the following relation is valid:*

$$N_{i,m}(x) = \begin{cases} \hat{N}_{i,m}(x) & i \leq l-m \\ \dfrac{\hat{t} - \hat{t}_i}{\hat{t}_{i+m} - \hat{t}_i}\hat{N}_{i,m}(x) + \dfrac{\hat{t}_{i+m+1} - \hat{t}}{\hat{t}_{i+m+1} - \hat{t}_{i+1}}\hat{N}_{i+1,m}(x) & l-m+1 \leq i \leq l \\ \hat{N}_{i+1,m}(x) & i \geq l-1 \end{cases}$$

Coefficients of the new function are modified according to the following rule:

$$\hat{c}_i = \begin{cases} c_i & i \leq l - m + 1 \\ \lambda_i c_i + (1 - \lambda_i) c_{i-1} & l - m + 2 \leq i \leq l \\ c_{i-1} & i \geq l + 1 \end{cases}$$

where $\lambda_i = \dfrac{\hat{t} - \hat{t}_i}{\hat{t_{i+m}} - \hat{t}_i}$.

Degree Elevation The Degree Elevation algorithm represents $f(x)$ of order m as a function $\bar{f}(x)$ of order $m + 1$ with the same shape. Coefficients of the new function are modified according to the following rule:

$$\begin{cases} \bar{c_0} = c_0 \\ \bar{c_i} = \dfrac{c_i(n + 1 - i) + ic_{i-1}}{n + 1} & i = 1, \ldots, n \\ \bar{c}_{n+1} = c_n \end{cases}$$

A NURBS curve in $\mathbb{R}^n$ in parametric form is a vectorial function with n components that are NURBS functions. In particular:

Definition 1.21. *A* NURBS curve $c(t) \in \mathbb{R}^n$ *can be expressed as follows:*

$$c(t) = (c_1(t), \ldots, c_n(t)) = \sum_{i=1}^{m+K} P_i R_{i,m}(t) = \frac{\sum_{i=1}^{m+K} w_i P_i N_{i,m}(t)}{\sum_{j=1}^{m+K} w_j N_{j,m}(t)}$$

where the parameter $t \in [a, b]$, $c_i(t)$ are NURBS functions defined in (1.3) and the $P_i \in \mathbb{R}^n$ are called control points. The ordered sequence of control points forms a control polygon.

Let consider $S_x = (\mathbb{P}_n, N, \Delta_x)$ and $S_y = (\mathbb{P}_m, M, \Delta_y)$ two spaces of monovariates B-Spline functions, defined respectively on $[a, b]$ and $[c, d]$ of order m and n, knot partitions $\Delta_x = \{x_i\}_{i=1 \ldots h}$ and $\Delta_y = \{y_i\}_{i=1 \ldots k}$, multiplicity vectors $N = (n_1, \ldots, n_h)$ and $M = (m_1, \ldots, m_k)$ associated with Δ_x e Δ_y, S_x and S_y have dimensions $n+H$ and $m+K$ respectively, where $H = \sum_{i=1}^{h} n_i$ and $K = \sum_{i=1}^{k} m_i$. The tensor product space of S_x and S_y, denoted by $S_x \otimes S_y$, with dimension $(n+H)(m+K)$ that can be written as

$$S_x \otimes S_y = (\mathbb{P}_{n,m}, N, M, \Delta_x \times \Delta_y).$$

Definition 1.22. *We define a* tensor product B-Spline function *associated with $\Delta_x \times \Delta_y$ as*

$$s(x,y) = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} c_{ij} N_{i,n}(x) N_{j,m}(y), \tag{1.4}$$

where $N_{i,n}(x)$ and $N_{j,m}(y)$ are the Normalized B-Spline basis functions of S_x and S_y, associated with the extended partitions $\Delta_u^* = \{u_i\}_{i=1\ldots 2n+H}$ and $\Delta_v^* = \{v_j\}_{j=1\ldots 2m+K}$ respectively.

Definition 1.23. *A* NURBS bivariate function *is defined as*

$$r(x,y) = \frac{\displaystyle\sum_{i=1}^{n+H} \sum_{j=1}^{m+K} c_{i,j} w_{i,j} N_{i,n}(x) N_{j,m}(y)}{\displaystyle\sum_{i=1}^{n+H} \sum_{j=1}^{m+K} w_{i,j} N_{i,n}(x) N_{j,m}(y)} \tag{1.5}$$

where $w_{i,j} > 0$.

This function can be written in this form:

$$r(x,y) = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} c_{i,j} R_{i,n,j,m}(x,y)$$

where $R_{i,n,j,m}(x,y)$ are the Bivariate Rational B-Spline basis functions defined as follows:

$$R_{i,n,j,m}(x,y) = \frac{w_{i,j}N_{i,n}(x)N_{j,m}(y)}{\sum_{i=1}^{n+H}\sum_{j=1}^{m+K} w_{i,j}N_{i,n}(x)N_{j,m}(y)}.$$

A NURBS surface can be represented using a 3D points grid (ξ_i, η_j, c_{ij}) with $i = 1, \ldots, n+H$ e $j = 1, \ldots, m+K$ where ξ_i and η_j are the Greville's abscissas of the knot vectors in u and v direction. If the grid is represented on the plane (ξ_i, η_j) with $i = 1, \ldots, n+H$ and $j = 1, \ldots, m+K$ we have the pre-image of the Control Grid.

Definition 1.24. *A* parametric NURBS surface *is a vector function $s(u,v) \in \mathbb{R}^3$ with components $x(u,v)$, $y(u,v)$ and $z(u,v)$ which are NURBS functions, defined in (1.5) belonging to the same space. A parametric NURBS surface is written as follows:*

$$s(u,v) = \sum_{i=1}^{n+H}\sum_{j=1}^{m+K} P_{i,j}R_{i,n,j,m}(u,v)$$

where $P_{ij} = (x_{ij}, y_{ij}, z_{ij}) \in \mathbb{R}^3$ are the control points.

An example of NURBS surface and its control grid is shown in Fig.3.1.

G^1 join between NURBS surfaces Using NURBS surfaces to construct complex surface objects requires to consider a smooth connection between the several surfaces approximating piecewise the object. Adjacent NURBS surfaces need to be joined with geometric continuity at the same order. According to Definition 1.13 two surfaces are joined with G^1 continuity along an edge e if there exists an equivalent reparametrization of the surfaces such that, along e, the two surfaces have the same tangent plane. For bicubic and biquartic surfaces, conditions for G^1 join are introduced in [21].
G^1 Join between NURBS surfaces can be performed with the algorithm described in [24].

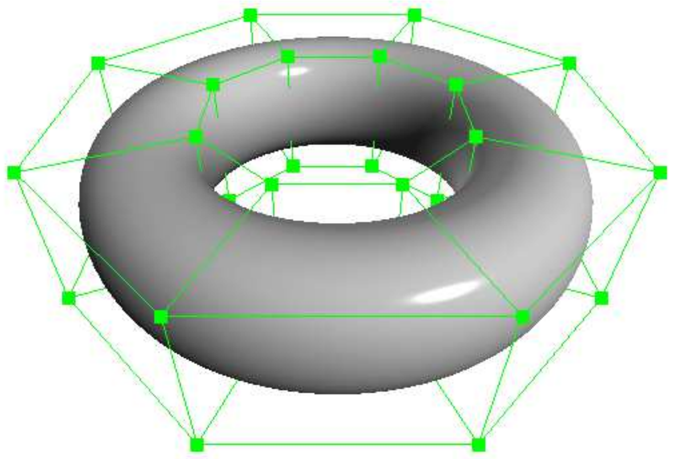


Figure 1.8: NURBS surface representing a Torus and its control mesh

Given two NURBS surfaces s_1 and s_2 with control points $d_{i,j}$ and $\tilde{d}_{i,j}$ and weights $w_{i,j}$ and $\tilde{w}_{i,j}$ respectively, we suppose to keep s_1 intact and modify the first and second lines of control points along the upper boundary strip of s_2 as follows, while keeping the other control points of s_2 unchanged. The following cases show how the new control points $\tilde{d}_{a,b}$ of s_2 are set as combination of control points d of s_1. The same is done for weights. Here we give the general conditions, which do not imply that s_1 and s_2 are joined along both entire edges.

1 At the lower-left corner position of the parametric domain of s_1, we set

$$
\begin{aligned}
\tilde{c}_{2r-2,0} &= \frac{2w_{0,0}c_{0,0} - w_{1,0}c_{1,0}}{2w_{0,0} - w_{1,0}} \\
\tilde{w}_{2r-2,0} &= 2w_{0,0} - w_{1,0} \\
\tilde{c}_{2r-2,1} &= \frac{2(w_{0,0}c_{0,0} - w_{0,1}c_{0,1}) - (2w_{1,0}c_{1,0} - w_{1,1}c_{1,1})}{2(2w_{0,0} - w_{0,1}) - (2w_{1,0} - w_{1,1})} \\
\tilde{w}_{2r-2,1} &= 2(2w_{0,0} - w_{0,1}) - (2w_{1,0} - w_{1,1})
\end{aligned}
$$

2 In the interior corresponding m pairs of boundary patches of the parametric domains of s_1 and s_2, we set

$$\tilde{c}_{2r-2+k,0} = c_{k,0}$$

$$\begin{aligned}
\tilde{w}_{2r-2+k,0} &= w_{k,0} \\
\tilde{c}_{2r-2+k,1} &= \frac{2(w_{k,0}c_{k,0} - w_{k,1}c_{k,1})}{2w_{k,0} - w_{k,1}} \\
\tilde{w}_{2r-2+k,1} &= 2w_{k,0} - w_{k,1}
\end{aligned}$$

with $k = 1, \dots, 2m$.

3 At the lower-right corner position of the parametric domain of $s1$, we set

$$\begin{aligned}
\tilde{c}_{2(r+m-1),0} &= \frac{2w_{2m+1,0}c_{2m+1,0} - w_{2m,0}c_{2m,0}}{2w_{2m+1,0} - w_{2m,0}} \\
\tilde{w}_{2(r+m-1),0} &= 2w_{2m+1,0} - w_{2m,0} \\
\tilde{c}_{2(r+m-1),1} &= \frac{2(w_{2m+1,0}c_{2m+1,0} - w_{2m+1,1}c_{2m+1,1}) - (2w_{2m,0}c_{2m,0} - w_{2m,1}c_{2m,1})}{2(2w_{2m+1,0} - w_{2m+1,1}) - (2w_{2m,0} - w_{2m,1})} \\
\tilde{w}_{2(r+m-1),1} &= 2(2w_{0,0} - w_{0,1}) - (2w_{1,0} - w_{1,1})
\end{aligned}$$

With these conditions s_1 and s_2 are G^1 connected.
According to literature, there are other algorithms that can be applied in order to solve this problem (i.e. [21]).

1.6 Polygonal Meshes

Polygonal meshes have a central role in our proposal. Let us introduce some basic definitions.

Definition 1.25. *A* mesh *$M = (V, E, F)$ is a collection of vertices V, edges E and faces F that defines a surface or the shape of a polyhedral object in solid modeling [17].*

The faces of the Mesh usually consist of triangles, quadrilaterals, or other simple convex polygons, but may also be composed of more general concave polygons, or polygons with holes. In our work we do not impose that all the polygonal faces are planar.

A *vertex* $v \in V$ is a point in the 3D space with associated information such as color, normal vector and texture coordinates. An *edge* is a straight line that connects two different vertices. A *face* is a closed polygon bounded by a cycle of edges.
An edge is called *interior* if it belongs to at least two faces, otherwise it is an *exterior* or a *boundary* edge.
A vertex is called *interior* if it doesn't belongs to a boundary edge, otherwise it is an *exterior* or a *boundary* vertex.
A mesh is called *closed* if every edge is interior, otherwise it the mesh is called *open* or *with boundaries.*
Furthermore we distinguish between *low*, *medium* and *high* resolution meshes. Low resolution meshes have $|V| \leq 100$, medium resolution meshes have $100 < |V| < 10000$ and high resolution meshes have $|V| \geq 10000$.
A *triangle mesh* is a mesh in which all faces are triangles, while a quad mesh is a mesh in which all faces are quadrilaterals. A *quad-dominant mesh* is a mesh in which the majority of faces are quadrilaterals, while there may be a small fraction of non-quadrilateral faces, typically triangles and/or pentagons.
The number of edges incident to a vertex is called *valence* or *degree* of the vertex.
Every mesh has an ideal valence, which is the valence of the *regular* vertices. All the vertices with a valence different from the ideal valence are called *extraordinary vertices* (EV).
For a triangular mesh an EV is an interior vertex with valence different from 6. Fig.1.9(b) shows a triangular closed mesh with EV colored in red and two of the non-EV in blue. If the triangular mesh is open, a boundary vertex with valence different from 4 is considered an EV. Fig.1.9(a) shows a quad closed mesh with EV in red and some of the non-EV in blue.
For a quad mesh the EV is an interior vertex of the mesh with valence different from 4. If the quad mesh is open, a boundary vertex with valence different from 3 is considered an EV.

According to [17], a mesh is said to be *regular* if the ideal vertex valence is

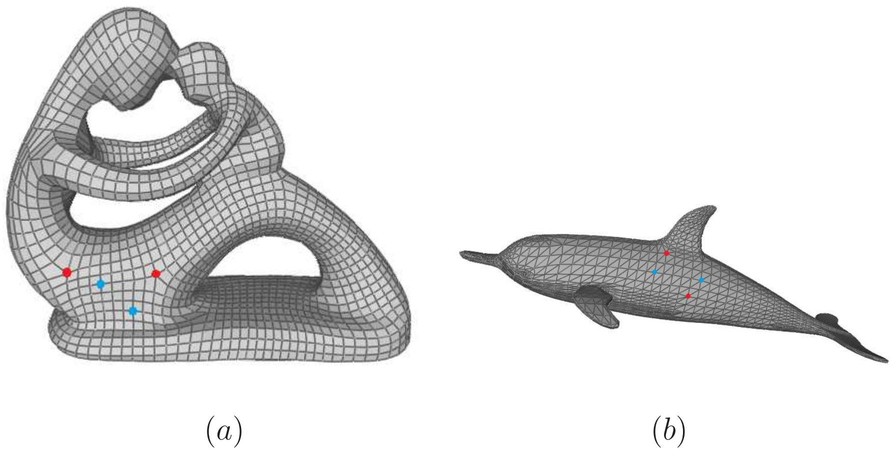


Figure 1.9: Examples of structured meshes: (a) quad mesh representing the Fertility statue, (b) triangular mesh representing a dolphin

maintained for all internal vertices of the model. For a quad mesh, a completely regular mesh is defined to be the one where all vertices have valence 4. This constraint is difficult, often impossible, to satisfy, as only genus-1 (toroidal) models can be described as a regular mesh.
A mesh is said to be *valence semi-regular* if there is a restricted number of extraordinary vertices. The EVs define the boundary curves of a coarse segmentation of the model. Each of the coarse regions is described by a mesh. Valence semi-regular meshes are able to describe surface models of arbitrary genus, while exhibiting the structural regularity that facilitates many geometric processing algorithms.
A mesh is said to be *unstructured* if a large fraction of its vertices have valence different from the ideal valence.
Fig.1.9 shows two examples of valence semi-regular meshes representing the Fertility statue and a dolphin, while Fig.1.10 shows two examples of unstructured meshes representing a femur and a bunny.
Moreover, we distinguish between *conforming* and *non-conforming* meshes.

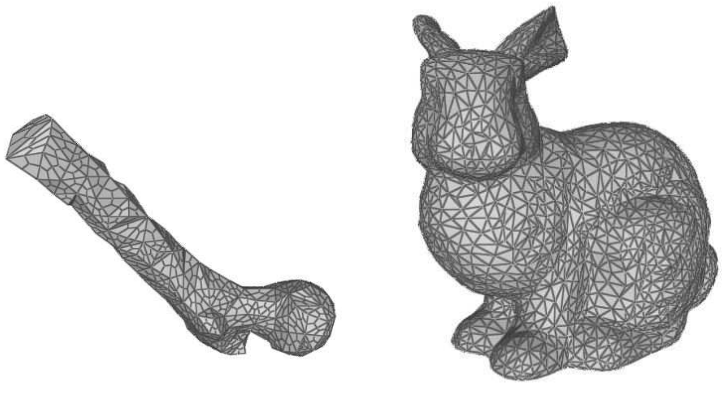


Figure 1.10: Examples of unstructured meshes: (a) quad mesh representing a femur, (b) triangular mesh representing a bunny

Conforming meshes have the property that any two faces may share either a single vertex, or an entire common edge. Non-conforming meshes do not respect this property.

T-meshes, are a special case of non-conforming meshes: in a T-mesh, there

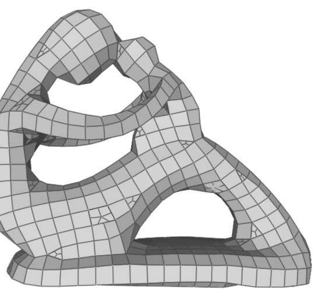


Figure 1.11: Examples of T-mesh representing the fertility statue

may exists an edge e of a face f that coincides with a chain of edges of two or more faces glued to f along e. All the internal vertices of such a chain, that split e, are called T-junctions of the mesh. An example is illustrated in Fig.1.11.

A mesh can be represented in different ways. The minimal data structure required is the Face-Vertex mesh which consists of a simple list of vertices, and a set of polygons. More sophisticated but useful for processing mesh is the winged-edge data structure, analogous to the winged-edge data structure described for B-Rep. Winged-edge meshes allow constant time traversal of the surface, but with higher storage requirements. There exists many different standard file formats to store mesh data such as *.obj,.stl,.ply* and *.3ds*.

Chapter 2

Extended Solid Modeling System

Our aim is to close the gap between parametric and discrete geometry for representing solid objects. We introduce a new solid modeling system that includes a new paradigm to represent and edit solid models. In particular, we extend a standard B-Rep scheme in order to make analytical surfaces and polygonal meshes coexist.
In the first part of this chapter we describe the main aspects of a solid modeling system. Then we introduce the concept of Mesh-Face and define the Extended B-Rep scheme. These are the basic concepts of the new Extended Solid Modeling System that we propose in this thesis. In the second part of this chapter we investigate the mathematical foundations necessary for our new system, with particular attention to the smooth connection between Mesh-Faces and NURBS or analytic surfaces. In the last part of the chapter we provide a high-level overview of methods realized for our Extended Solid Modeling System.

2.1 Solid Modeling Systems

A solid modeling system, often called solid modeler, is a computer program that provides facilities for storing and manipulating data structures that represent the geometry of solid objects or assemblies.

Tipically it allows to select and manipulate modeling primitives, such as lines, cubes and cylinders, and invokes modeling operations to combine these primitives into more elaborate representations.
The structure of a solid modeling system can be subdivided in three sections: the representation data structure, the mathematical foundations and the algorithms necessary for the applications. The representation data structure is the scheme used to represent a physical object. In chapter 1 we introduced the most important representation schemes that supply to the specific need for informational completeness in mechanical geometric modeling systems [45].
The mathematical foundations are all those abstract concepts that allow to idealize and approximate a physical object. These abstractions and idealization, which involves geometric representation of the shape and approximation models, allow to consider the object as a perfect and homogeneous 3D point set, ignoring internal structures and boundary imperfections. Moreover continuity between geometric representations is considered in order to join solid objects. These are the basis for the algorithms necessary to model the object. The algorithms are tools necessary to represent, modify and investigate solid objects.

In our work we consider a solid modeling system with a Boundary Representation scheme (B-Rep) and extend it in order to manage both analytical surfaces and polygonal meshes. In the next section we introduce the new data structure of our Extended Solid Modeling System: the Mesh-Face.

2.2 Mesh-Face

In Chapter 1 we introduced the main characteristics of a B-Rep scheme. We introduced the “face” topological element explaining that the geometric entity associated with this element could be a plane, an analytic surface or a NURBS surface. The Extended B-Rep exploits all these kinds of primitives

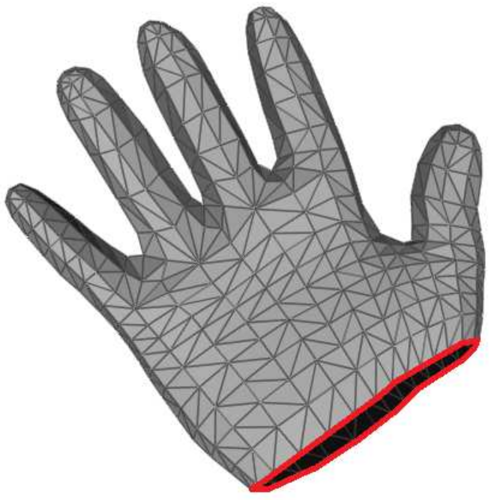


Figure 2.1: Example of Mesh-Face representing a hand

as a topological face and furthermore it considers the new primitive Mesh-Face.

A Mesh-Face consists in a mesh of polygonal facets with boundaries associated with a single face of a B-Rep representing a solid object. A Mesh-Face can represent both the boundary of a solid object and just a part of it, that we call submesh.

The proposed scheme includes both the trivial cases, where the solid is described by only one Mesh-Face, and the more general cases, where the Mesh-Face, delimited by a closed polyline, represents one face of the EB-Rep. An example of Mesh-Face is shown in Fig.2.1 with its boundary polyline colored in red. The Mesh-Face is handled exactly as a standard face in a B-Rep data structure. Therefore the loop of the face is defined by the polygonal boundary of the Mesh-Face.

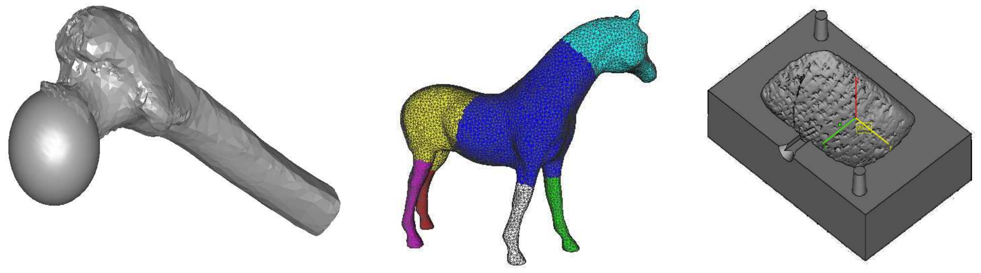
2.3 Extended B-Rep

In many situations a solid can be more intuitively described by meshes for some parts of it and by parametric or exact surfaces for other parts. In these cases, we introduce a suitable representation, named Extended B-Rep (EB-Rep), that aims at closing the gap between parametric and discrete geometry in the representation of solid objects.

Definition 2.1 (Extended B-Rep)**.** *An* Extended B-Rep *is a representation scheme*

$$B_e = (G_e, T)$$

where the geometry is described by $G_e = (V, E, F_e)$ *and the set of the faces* F_e *admits also Mesh-Faces.*



(a) (b) (c)

Figure 2.2: Three examples of Extended B-Rep: (a) femur composed by a Mesh-Face with a NURBS sphere, (b) Horse represented by Mesh-Faces, (c) Mold model obtained by a point cloud triangulation composed by a Mesh-Face and analytical surfaces

This new structure has to maintain the same properties, in particular the same topology T, and to provide the same tools of the standard B-Rep, while holding the new potential for Mesh-Face primitives.
Three examples are shown in Fig.2.2. Fig.2.2(a) shows an example of EB-Rep composed of two faces: a NURBS spherical face and a Mesh-Face representing

a part of a femur. Fig.2.2(b) shows a particular EB-Rep model composed of multiple Mesh-Faces bounded by polylines. Fig.2.2(c) illustrates an example of a mold where the body is described by planar, analytic and NURBS surfaces, while the cavity is a Mesh-Face representing the object to be molded. A Mesh-face represents the triangulation of the point cloud of the object.

Mesh-Faces and EB-Reps are the basis of our innovative Extended Solid Modeling System, where meshes and NURBS surfaces coexist.
In order to realize this new system it is necessary to handle the interaction between classic B-Rep entities and meshes. Most of the tools are directly inherited from the standard solid modeling systems that use analytic and NURBS entities. Also polylines are handled in classic B-Rep systems. Instead, the notion of continuity between Mesh-Face and NURBS entities has to be investigated, because it is necessary to define a new concept of continuity between smooth and discrete entities. Concerning the tools for the EB-Rep modeling system the Boolean Operations need particular care to be managed, considering in particular the surface-to-surface intersection problem. In the next sections we investigate and introduce possible solutions for these two problems.

2.4 Continuity for Extended B-Rep

For Extended B-Rep models it is impossible to create an exact smooth join between a Mesh-Face and a NURBS surface. Meshes are piecewise linear approximations, under a given tolerance, of analytic surfaces, thus it is only possible to give some less restrictive conditions in order to obtain a join that is smooth under a given tolerance.
We present an alternative definition of continuity between discrete and continuous entities: the Approximated Geometric (AG) continuity, similarly introduced in [44].

Definition 2.2 (AG^0 continuity)**.** *Given a surface $s(u, v)$ with a boundary curve $c(t)$ and a mesh M with a boundary polyline p, we say that s and M join with AG^0* ***continuity*** *along the boundaries c and p, according to a given tolerance tol, if and only if:*

$$\delta_H(p, c) = \max(\bar{\delta}_H(p, c), \bar{\delta}_H(c, p)) \ < \ tol \tag{2.1}$$

where

$$\bar{\delta}_H(A, B) = \max_{a \in A} \min_{b \in B} d(a, b) \tag{2.2}$$

$\delta_H(A, B)$ is called the bivariate Hausdorff distance between curves, while $\bar{\delta}_H(A, B)$ is the univariate Hausdorff distance between curves and $d(a, b)$ is the Euclidean distance.

This definition implies that the distance between all points of p from c and the distance of all points of c from p has to be $< tol$. Fig.2.3(a) shows an example of the described condition: in this case the p is shorter than c, so $\bar{\delta}_H(p, c) < tol$ but $\bar{\delta}_H(c, p) > tol$, since $d(c_0, p) > tol$.
Moreover, for every point $p_i \in p$ there is a point $c^* \in c$ such that $d(p_i, c^*) = \bar{\delta}_H(p_i, c)$ and for every point $c_i \in c$ there is a point $p^* \in p$ such that $d(c_i, p^*) = \bar{\delta}_H(c_i, p)$. We observe that if c^* is the nearest point to p_i on c, this does not imply that p_i is the nearest point to c^* on p. Fig.2.3(b) shows an example of the previous statement in which c^* is the nearest point to p_2, but p_2 is not the nearest point to c^*.

Definition 2.3 (AG^1 continuity)**.** *Given a surface $s(u, v)$ with a boundary curve $c(t)$ and a mesh M bounded by a polyline p. Assume M and s join with AG^0 continuity along c and p respectively. We say that s and M join with AG^1* ***continuity*** *along c and p respectively, according to a given tolerance tol_1, if and only if:*

$$\delta_{H\alpha}(p, c) = \max(\bar{\delta}_{H\alpha}(p, c), \bar{\delta}_{H\alpha}(c, p)) \ < \ tol_1 \tag{2.3}$$

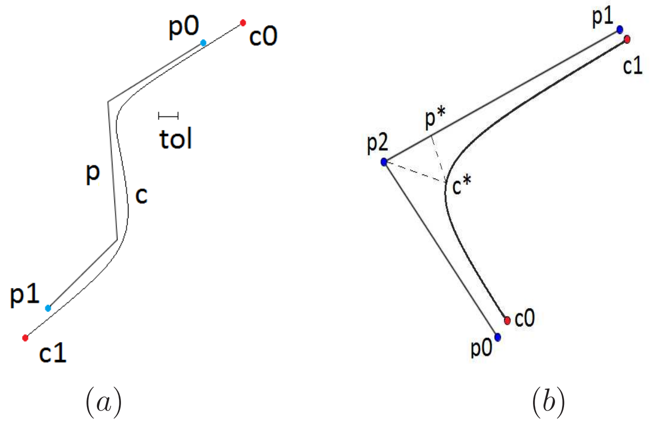


Figure 2.3: Curve c and polyline p for the definition of AG^0 Continuity

where

$$\bar{\delta}_{H\alpha}(A,B) = \max_{a\in A} \vec{n}_A|_a \cdot \vec{n}_B|_{b^*} \quad \textit{with} \quad b^* \in B \; s.t. \; b^* = argmin_{b\in B} d(a,b) \tag{2.4}$$

where $\vec{n}_A|_a$ is the normal vector to A in a, $\delta_{H\alpha}(A,B)$ is the bivariate angular Hausdorff distance and $\bar{\delta}_{H\alpha}(A,B)$ is the univariate angular Hausdorff distance.

This definition implies that the angle between the normal vectors at two closest points respectively on c and on p is smaller than a given tolerance. The univariate angular Hausdorff distance is computed considering for every point $a \in A$, the nearest point $b \in B$. Considering the previous remark and Fig.2.3(b), we observe that the couple of points (a,b) used to compute $\bar{\delta}_{H\alpha}(a,B)$ is not necessarily the couple of points used to compute $\bar{\delta}_{H\alpha}(b,A)$.

Definition 2.4 (G^1-AE continuity)**.** *Let a surface $s(u,v)$ with a boundary curve $c(t)$ and a mesh M bounded by a polyline p with vertices $p_1,\ldots,p_n$ be given. Let M and s be joined C^0 along p and c. M and s join with G^1-* ***Almost Everywhere Continuity*** *along c and p if and only if they are G^1 connected along all points of c except at $p_1,\ldots,p_n$, where c is possibly not differentiable.*

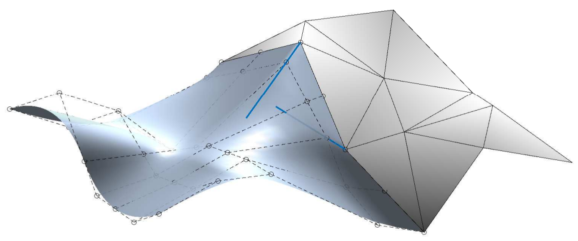


Figure 2.4: An example of G^1-Almost-Everywhere Continuity

An example of G^1-AE Continuity is illustrated in Fig.2.4 where a NURBS surface, on the left, and a mesh on the right, join with G^1-AE continuity. The surfaces have C^0 continuity along the boundary curve and G^1 continuity everywhere except at the vertices of the mesh on the boundary polyline.

These definitions allow us to introduce less restrictive connection conditions for a join between a Mesh-Face and a NURBS. In particular the definitions of AG^0 and AG^1 coincide with the main idea of numerical approximation. All the main numerical algorithms for solid modeling are based on a given tolerance, this because the use of finite numbers introduces errors due to the impossibility to exactly represent a real number on a machine with finite memory. A lot of applications, involving 3D scanners for example, determine the quality of an object considering the given tolerance of the 3D scanner equipment. Therefore, concepts such as AG^0 and AG^1 continuity are commonly used in all numerical algorithms and in all those applications that use these algorithms.
Instead, the last definition is the most intuitive one, also if the result surface is not differentiable on its boundary. In this case the join is smooth except for a finite number of points and in these points the two tangent vectors have a distance angle that is related with the dihedral angle between all adiacent polygons.

2.5 Methods

In this section our goal is to provide a high-level overview of typical methods involved in solid modeling which can be generalized for our Extended Solid Modeling System. In particular we focus on Boolean Operations, Cutting and Join Operation. These tools play a fundamental role in solid modeling. Boolean operations allow to create complex objects from simple primitives. Cutting operation allows to model a solid object by using surfaces. Join operation is a fundamental tool that allows to build a new model from two or more different models by matching them along boundaries.

2.5.1 Boolean Operation with Extended B-Rep

Boolean Operations (BO) are the basic tools used to model a solid object. The combination of Union, Intersection and Difference operations on primitive objects allow us to create complex objects.
In solid modeling, the set-theoretic Boolean Operations are substituted by the *Regularized Boolean Operations* (RBO), in which the result is the closure of the BO between the interior of the two solids. This is done in order to eliminate the remaining lower-dimensional structures. Given two solids A and B and a BO op, its corresponding RBO, denoted by $op*$ is defined as

$$A\ op*\ \ B = cl(\imath A\ op\ \imath B) \tag{2.5}$$

where $cl(A)$ denotes the closure of A. An example is illustrated in Fig.2.5 in which it is possible to see the difference between a classic BO and a RBO. Fig.2.5(a) shows the two solids A and B, Fig.2.5(b) shows the BO between A and B that produces a non-manifold object with an isolated surface. Fig.2.5(c-d) show respectively the interior of the RBO operation between A and B and the result of RBO, that prevents the situation illustrated in Fig.2.5(b) creating in every case a manifold solid. In a solid modeling system, a RBO between two solids A and B is performed determining the boundary

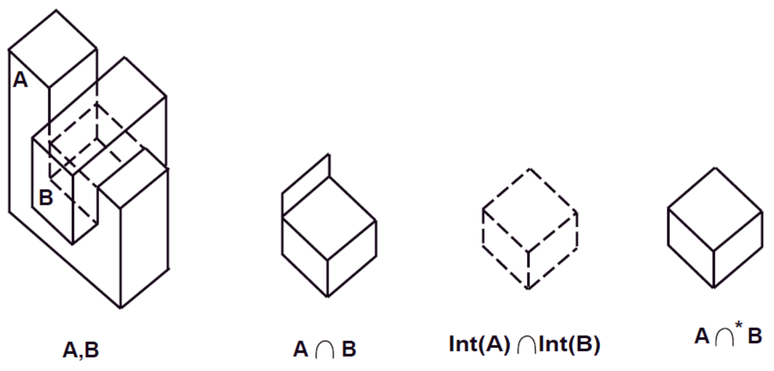


Figure 2.5: Difference between Boolean Operation and Regularized Boolean Operation

$\flat$ of the result solid C. In particular we define:

$$\flat(A \cap B) = (\flat A \cap \imath B) \cup (\flat B \cap \imath A) \tag{2.6}$$

$$\flat(A \cup B) = (\flat A \cap cB) \cup (\flat B \cap cA) \tag{2.7}$$

$$\flat(A \setminus B) = (\flat A \cap cB) \cup (\flat B \cap \imath A) \tag{2.8}$$

where $\imath A$ is the interior of A and cA is the external part of A. The boundary $\flat$ of the resulting solid is computed applying a surface-surface intersection algorithm that determines all the intersections between surfaces of A and B. Then an algorithm is applied to create the B-Rep structure of the object.
In an Extended Solid Modeling System in which Mesh-Faces are managed it is necessary to distinguish between three possible cases:

. NURBS - NURBS intersection

. Mesh - Mesh intersection

. NURBS - Mesh intersection

The first case is the classic surface-surface intersection between two NURBS

surfaces. This problem has been investigated since 1987 and two of the most famous solutions are proposed in [10] and [31]. Given two surfaces in $\mathbb{R}^3$, the intersection can be either a set of isolated points, a set of curves, a set of overlapping surfaces or any combination of these cases. In [10] a marching method is applied to compute the intersection, while in [31] a loop detection, that recursively subdivides two surfaces until no surface patches intersect in a closed loop, is proposed. Open source libraries as OpenCascade [5] and CGAL [3] and other CAD systems use a similar algorithm to compute RBO between solid objects represented with B-Reps.
The Mesh-Mesh intersection is a case managed in the solid systems that use only meshes. As an example, Carve library [1] realizes boolean operations between meshes. The structure of the surface-surface intersection algorithm is analogous to the one described for NURBS surfaces. Once the intersections are computed, a classification is performed in order to create the result mesh. We refer to [23] where the boolean operations between meshes in the system *DesignBase* are described in details. Once the intersection lines and points are computed, the included parts are removed and the two solids are joined creating the final mesh.

The NURBS-Mesh intersection is the novel case that has to be considered in order to perform RBO in an Extended Solid Modeling System. In this case intersection between two entities is computed considering the AG^0 continuity. In particular, the intersection curves between a Mesh-Face M and a NURBS surface s are respectively a polyline p that bounds the Mesh-Face and a NURBS curve c that bounds the NURBS surface. The result surfaces are a trimmed NURBS and trimmed Mesh-Face. An example is illustrated in Fig.2.6 where a NURBS and a triangular mesh are intersected. The intersection polyline is marked by points in red, where c and p intersect each other. The polyline is the boundary polyline of the Mesh-Face illustrated. As we can notice, the polyline is C^0 with the Mesh-Face, while is AG^0 with the NURBS surface. The precision of the intersection polyline respects the tolerance used in the surface-surface intersection algorithm and depends also

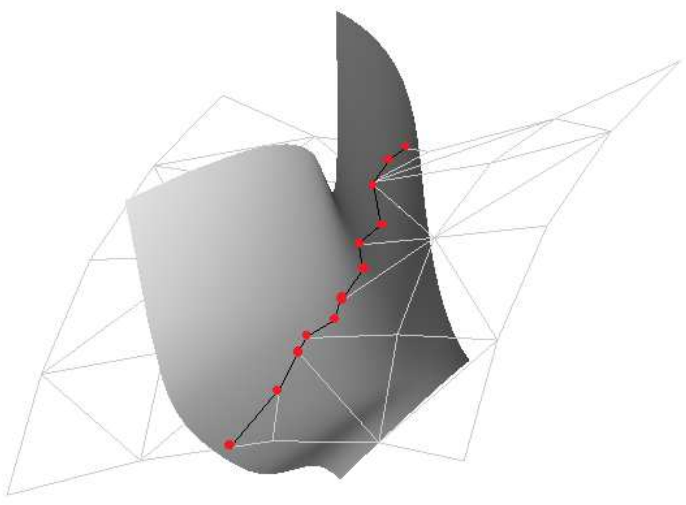


Figure 2.6: Intersection between a NURBS and a mesh

on the tolerance associated with the AG^0 continuity.

2.5.2 Cutting Operation

Cutting is an operation between a solid A and a surface s which results in two distinct solid components, A_t and A_b, that share a common face s. According to the normal of the surface, A_t or A_b is chosen. This tool plays a fundamental role in "Hybrid Solid Modeling Systems", which are systems that allow to create a solid object, with a freeform surface as boundary, modeling it with surfaces.
Actually all the most important industrial CAD system such as Inventor, SolidWorks and Catia use cutting operation in order to efficiently model a solid object. An example of multiple cutting operation is shown in Fig.2.7. Fig.2.7(a) shows the solid and the surfaces for cutting, Fig.2.7(b) illustrates how the solid is bounded by the surfaces, Fig.2.7(c) shows the final solid obtained. The solid is originally delimited by NURBS surfaces and is cut by NURBS surfaces.

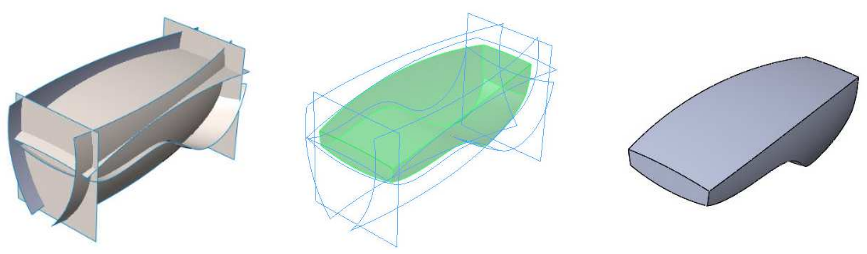


Figure 2.7: Example of Cutting operations with a classic B-Rep model and NURBS surfaces

In an Extended Solid Modeling System this tool allows to cut EB-Reps with both NURBS surfaces and Mesh-Faces. Result of this tool is an EB-Rep. In this case, similarly to the Boolean Operations introduced before, the surface-surface intersection between Mesh-Faces and NURBS surfaces has to be managed.

2.5.3 Join Operation

The Join Operation attaches two entities changing one or both entities. The possibility of modifying the entities makes this tool differ from Boolean Operation, in which both entities are fixed. We distinguish between 1-1 Face-Join operation, 1-n Face-Join and n-m Face-Join of open solids. 1-1 Face-Join matches two surfaces along an edge, creating a connected surface. 1-n Face-Join closes the hole of an open solid with a surface. n-m Face-Join of open solids matches two open solids closing a hole on both entities.
When a Join operation is performed it is necessary to specify a level of regularity along the boundary edges which determines the smoothness of the resulting surface or solid. It is possible to have C^0 or G^n regularity, with $n = 1, 2, \ldots$. In our work we consider only the C^0 and the G^1 cases.
In case of C^0 continuity, the join produces an object C that has points around the joining area in which is not differentiable, or rather that there exist points

that have two distinct tangent planes. Surfaces that are G^1 joined are differentiable with derivatives until first order along the joining boundary.
In an Extended Solid Modeling System these tools allow to join NURBS surfaces and Mesh-Faces. Results are EB-Rep solids. In this case it is important to manage the smooth joining between NURBS surfaces and Mesh-Faces according to the definitions of AG^1 and G^1-AE continuity.

2.6 A new I/O format to manage Extended B-Rep

The Extended Solid Modeling System introduced in this chapter manages both Mesh and NURBS entities and represents solids obtained by modeling these entities using an Extended B-Rep scheme. In order to import or to export this new kind of solids in which NURBS and meshes coexist it is necessary to realize an extension of the standard exchange format "$STEP$" (STandard for the Exchange of Product model data) [6]. That new format, called "Extended-STEP" allows to save an entire mesh in a Mesh-Face structure. In particular there are two new entities called "$MESH\_FACE$" and "$POLYGON\_FACE$" respectively stored in this format:

$$\#l = MESH\_FACE(nFaces, \#nf1, \#nf2, \ldots, \#nfn)$$

$$\#nf1 = POLYGON\_FACE(nVertices, \#nv1, \#nv2, \ldots, \#nvn)$$

$$\#nv1 = CARTESIAN\_POINT(' \ ', (x, y, z))$$

Introducing these new entities in the STEP format it is possible to represent Mesh-Faces and manage import and export of EB-Reps.

Chapter 3

EB-Rep form of a Valence Semi-Regular Mesh

An Extended B-Rep paradigm can be realized as a new data structure in an Extended Solid Modeling System. However, a most typical scenario could require the integration of the Mesh-Face primitive into an existing Solid Modeling System based on a classical B-Rep paradigm. In this case the data structure can not be modified and thus finding an alternative way to represent a mesh in a standard B-Rep data structure becomes necessary. In our work we realized such a system, based on B-Rep data structure, which manages both meshes and NURBS surfaces.
According to the literature, the most intuitive way to represent a mesh surface is to associate a plane with every face. This approach is implemented in all the common CAD systems in order to represent a solid model given a mesh representing its boundary. Alternatively, if the mesh is a quad mesh, we can associate a NURBS bilinear surface with every face. Both these methods allow us to realize the reverse process, returning to the original mesh after a given B-Rep processing, without losing any piece of information, but they require a number of B-Rep faces equal to the number of faces in the original mesh. Considering that, usually, the meshes are defined by milions of facets, both these methods would lead to an inefficient implementation of an EB-

Rep scheme in a standard Solid Modeling System.
In this chapter we propose a new approach, suitable for valence semi-regular quad meshes. In the second part of the chapter we extend this method to triangular valence semi-regular meshes. The representation of general unstructured meshes as EB-Rep is discussed in chapter 4.
All the methods proposed in this work have been implemented in our system, based on OpenCascade library, in order to verify the validity of our proposals and provide examples. Images illustrated in this chapter and in the following are realized with our OpenCascade system. Note that a EB-Rep face is graphically represented with a 3×3 grid of curves ($3 \times 3gc$) as shown in Fig.3.1. The boundary curves of the grid correspond exactly to the boundary curves of the geometry associated with the topological face, while the inside curves are added for visualization purposes.

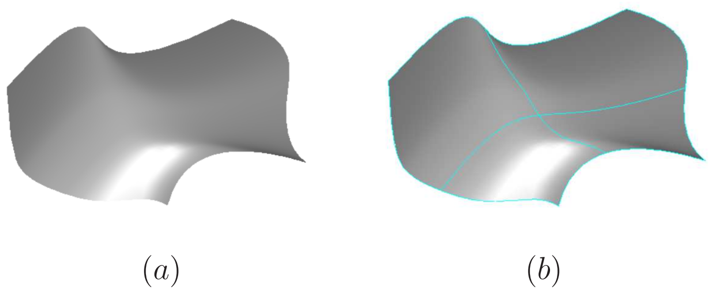


Figure 3.1: a) NURBS surface b) NURBS surface represented with our Open-Cascade system

3.1 Quadrilateral mesh decomposition in rectangular patches

Our approach describes a quad mesh, representing the boundary of a solid object, by an EB-Rep with faces described by a low number of NURBS surfaces, without losing any information.

In the simplest case in which the mesh is regular, then it can be represented by a unique NURBS bilinear surface whose control points are the mesh vertices. In the more general case, first we need to subdivide the mesh into submeshes with rectangular topology and then to associate a bilinear NURBS surface with each submesh.
Assuming to have a valence semi-regular mesh, we realized two methods for the mesh decomposition step:

. *Quad Mesh Patching*(QMP): Create rectangular patches without T junctions. More in detail, the result B-Rep structure has only faces delimited by four edges and four vertices. Every vertex is a corner vertex for every face that contains it.

. *Quad Mesh T-Patching*(QMTP): Create rectangular patches with T junctions.

Given a quadrilateral mesh $M = (V, E, F)$ with or without boundary, the methods create an EB-Rep $B_e = (G_e, T)$, $T = (V_T, E_T, F_T)$ where:

. $V_T \subset V$

. $E_T = \{\tilde{e} \in E_T \mid \tilde{e} = \bigcup e_{I_j}, \quad e_{I_j} \in E, \quad I_j \in \{1, \ldots, |E|\}\}$

. $F_T = \{\tilde{f} \in F_T \mid \tilde{f} = \bigcup f_{I_j}, \quad f_{I_j} \in F, \quad I_j \in \{1, \ldots, |F|\}\}$

The EB-Rep topological structure T has vertices that are vertices of the mesh M and edges that are obtained gluing edges in E of the original mesh M. With any new face in F_T is associated a submesh with rectangular topology formed by adjacent faces in F. B_e is the B-Rep description of the original mesh M. In order to better describe the methods we recall the following definitions:

. $EV \subset V$ the set of extraordinary vertices of V. These are all internal vertices with valence different from 4 and all boundary vertices with valence different from 3.

. $BV \subset V$ the set of boundary vertices of V that are not in EV.

. $NV = V \setminus (EV \cup BV)$ the set of inner, non-extraordinary vertices.

Both the QMP and QMTP methods can be subdivided in the following steps:

Step 1: Select the set of extraordinary vertices EV.

Step 2: Create the set of edges E_T.

Step 3: Create the set of faces F_T from vertices and edges of the new mesh T.

Step 1. Collect all the extraordinary vertices $ev_i \in EV$ of the original mesh M. If M has all extraordinary vertices ($NV \equiv \{\emptyset\}$) the method is stopped because there is no possibility to decrease the number of faces, that is $|F_T| = |F|$. A NURBS patch is associated with every face of F_T. If no extraordinary vertex is found ($EV \cup BV \equiv \{\emptyset\}$), the mesh is represented with a single rectangular bilinear NURBS patch. In this case the vertex chosen as origin of the patch is freely chosen.

Step 2. From every edge $e \in E$ starting from every extraordinary vertex $ev_i \in EV$ a polyline $\tilde{p}$ is traced composed of edges in E. $\tilde{p}$ is called *straight edge* and it is built as follows:

. $\tilde{p}$ starts from $ev_i \in EV$ and ends when either another extraordinary vertex $ev_j \in EV$, or a boundary vertex $bv \in BV$ or $\tilde{e}_i \in E_T$ are encountered. In the last two cases, bv and $v_k \in V$ such that $v_k = \tilde{p} \cap \tilde{e}_i$ are added to V_T and marked as 'visited'.

. $\tilde{p}$ is built following a *straight line* determined from $v_i \in V$. In particular, when an ordinary vertex $v_i \in V$ is visited, the list of edges incident this vertex is read and the algorithm proceeds in the straight direction. That is the new edge of the straight line and is determined as the second edge of the list after the edge considered.

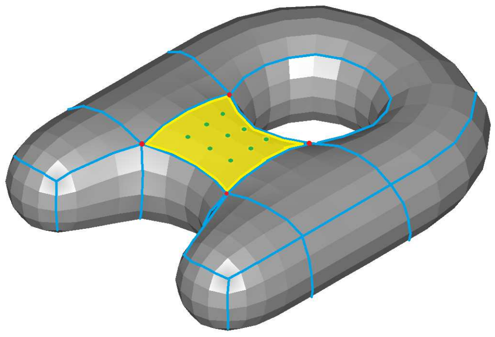


Figure 3.2: Result of the Step 2 for the decomposition of the "A" quad-mesh.

. If $\tilde{p}$ arrives at a visited inner vertex $v_k \in V$ it means that $\exists\ \tilde{e}_i$ such that $\tilde{p} \cap \tilde{e}_i \neq \emptyset$. In this situation we have two possibilities:

  1 for QMTP a new T-vertex $v_t = v_k$ is created and v_t is added to V_T, $\tilde{e}_i$ is split, a new edge $\tilde{e}_j$ delimited by v_t and ev_i is created and the algorithm goes to the next step.

  2 for QMP a v_k is added to V_T, $\tilde{e}_i$ is split, a new edge $\tilde{e}_j$ delimited by v_t and ev_i is created and a new edge $\tilde{e}_k$, that has v_k as starting vertex is created.

. If $\tilde{p}$ arrives at a visited vertex on the boundary, we have two possibilities:

  1 If this vertex is contained in a boundary edge of the new mesh T, the edge is split.

  2 If this vertex does not belong to any edge of the new mesh T, the vertex is added to the list of the extraordinary vertices.

An example is illustrated in Fig.3.2 where the "A" quad-mesh is decomposed by applying the QMP method. The extraordinary vertices in EV are colored in red and the straight line traced from the extraordinary vertices are colored in blue. The initial "*A*" mesh $M = (512, 1024, 512)$ is represented with an EB-Rep whose topological structure is $T = (24, 48, 24)$.

Step 3. The NURBS faces are created in the last step of both the methods. Starting from V_T and E_T this step first determines the vertices for each rectangular face $\tilde{f} \in F_T$ and then builds the new NURBS surface considering these vertices as control points. In particular:

. *Determine the boundary edges for each face in* F_T. For every new vertex $\tilde{v} \in V_T$ all the new incident edges $\tilde{e} \in E_T$ are considered and saved using a counterclockwise order. Every couple of consecutive edges $(\tilde{e}_i,\tilde{e}_j)$ represent a corner of new face $\tilde{f}$ (with the exception of T-junctions), thus determinating the first three corner vertices of $\tilde{f}$. Let us denote by $\tilde{e}_i = (x, x_i)$ and $\tilde{e}_j = (x, x_j)$. Then a searching algorithm is used to compare the other couples of consecutive edges incident to x_i and x_j to find the common vertex, that is the fourth corner vertex. Finally the face boundary is built.

. *NURBS patching.* The bilinear NURBS surface is defined by building the grid of control points from the vertices of M following the 'straight edges' and the mesh structure. The control points of the NURBS patch are determined considering the regular structure of the sub-meshes.

Considering the decomposition of the "A" mesh, in Fig.3.2, a new NURBS face, overimposed in yellow, is created. The control points of the NURBS bilinear surface are the control points of the boundary curves and the vertices of M, marked in green.

Examples We tested the QMP and QMTP methods by implementing the two algorithms in our OpenCascade platform. In Fig.3.3 some examples of closed valence semi-regular quad meshes efficiently represented as an EB-Rep with bilinear NURBS surface patching are shown.

We also tested some open meshes, represented in Fig.3.4. The results obtained are reported in Table 3.1 and Table 3.2, respectively.

Observing both sets of examples, from Tables 3.1 and 3.2 we notice that

Mesh	$\|F\|$	$\|V\|$	$\|EV\|$	$\|F_T\|$ QMP	$\|V_T\|$ QMP	$\|F_T\|$ QMTP	$\|V_T\|$ QMTP
A	512	512	16	24	24	18	20
H Cube	80	66	26	72	58	64	58
Human	806	808	100	344	346	160	203
Tooth	1132	1134	16	30	32	28	32
Fertility	3357	3351	48	2271	2265	132	191

Table 3.1: EB-Rep form of Quadrilateral Meshes (for meshes illustrated in Fig.3.3). From left to right: Mesh Name, number of faces, vertices, Extraordinary Vertices of M; number of faces in T by QMP, number of vertices in T by QMP, number of faces in T by QMTP, number of vertices in T by QMTP

Mesh	$\|F\|$	$\|V\|$	$\|EV\|$	$\|F_T\|$ QMP	$\|V_T\|$ QMP	$\|F_T\|$ QMTP	$\|V_T\|$ QMTP
Pawn	148	154	5	5	9	5	9
Rocker Arm	161	172	22	75	85	44	63
Tube	240	263	5	15	20	15	20

Table 3.2: EB-Rep form of Quadrilateral Meshes (for meshes illustrated in Fig.3.4). From left to right: Mesh Name, number of faces, vertices, Extraordinary Vertices of M; number of faces in T by QMP, number of vertices in T by QMP, number of faces in T by QMTP, number of vertices in T by QMTP

the method QMTP, which admits T junctions, is obviously more efficient than QMP in terms of number of facettes required. Moreover, the number of faces and vertices of the EB-Rep depends on the number of extraordinary vertices EV in the mesh M. The smaller is the number of extraordinary vertices, the fewer is the number of NURBS patches necessary to give an efficient representation of the mesh. The two closed meshes representing the "A" ($|F| = 512$) and the tooth ($|F| = 1132$) have a small number of extraordinary vertices ($|EV| = 16$). Their EB-Rep form needs a small number of

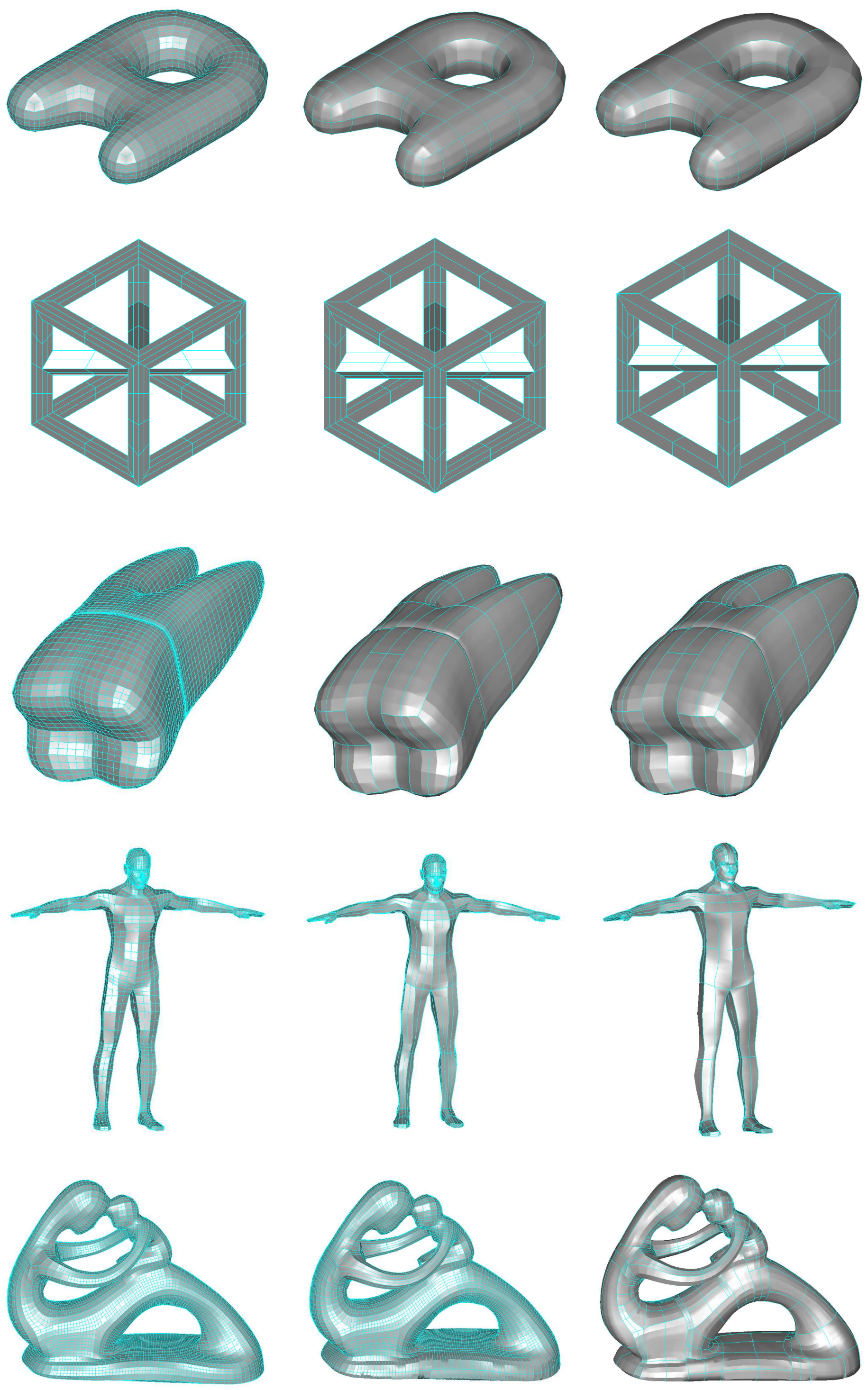


Figure 3.3: First Column: Initial Meshes M, Second Column: EB-Reps obtained by QMP, Third Column: EB-Reps obtained by QMTP. Faces are plotted with $3 \times 3gc$

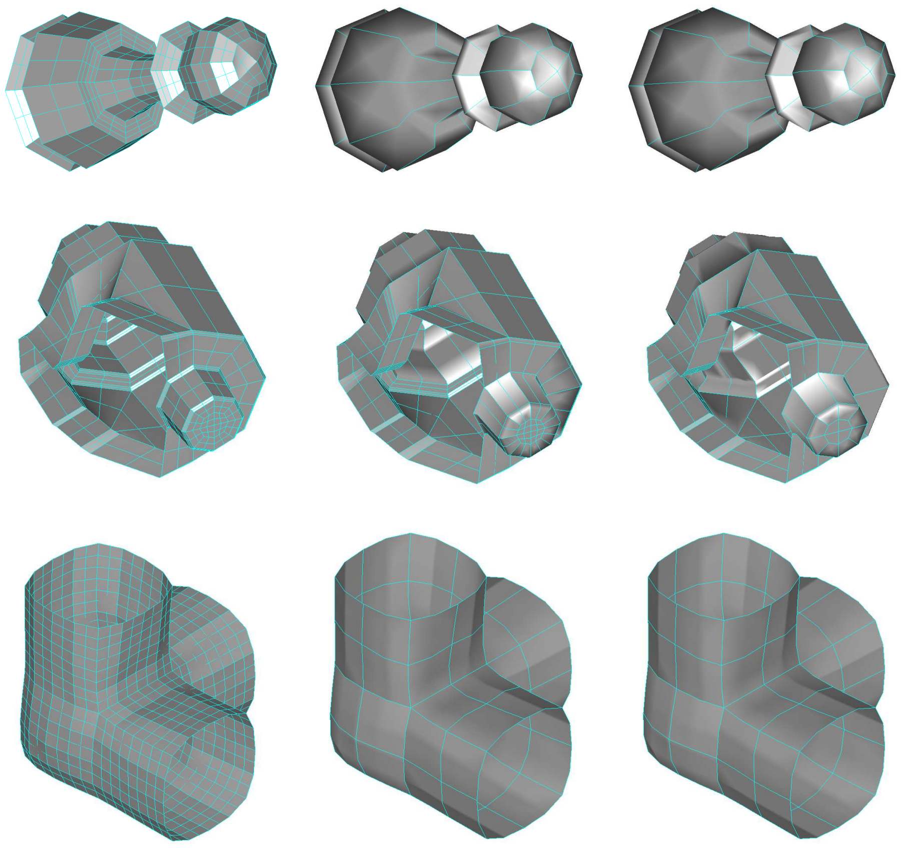


Figure 3.4: First Column: Initial open meshes M, Second Column: EB-Reps obtained by QMP, Third Column: EB-Reps obtained by QMTP. Faces are plotted with $3 \times 3gc$

NURBS patches, respectively 18 and 28 by QMTP. However, if the number of extraordinary vertices is higher the gain is lower.

3.2 Triangular to Quadrilateral mesh conversion

The QMP and QMTP methods introduced require a valence semi-regular quad mesh as input, thus if we need an efficient B-Rep description of a valence semi-regular triangular mesh we need first to convert it into a quadrilateral mesh without losing original information about the geometry of the mesh. To this aim, we considered and modified the method introduced in [47], which takes as input a triangular mesh and gives as output a quadrilateral mesh. The algorithm consists of four different steps:

Step 1: unification of 4-valence vertices

Step 2: matching of couples of triangles

Step 3: matching analysis

Step 4: quads subdivision of remaining triangles

We modified Step 2 by introducing a new condition to match couples of triangles in order to minimize the number of extraordinary vertices in the resulting quad mesh.

Step 1. The first step requires to determine all 4-valence internal vertices in the input triangular mesh. For every vertex v_i in V the 4 triangles sharing v_i are unified and subdivided into 4 quads. Fig.3.5 shows how Step 1 is performed. Vertices of the new quads are the middle points of the boundary edges and the vertices of the 4 triangles.

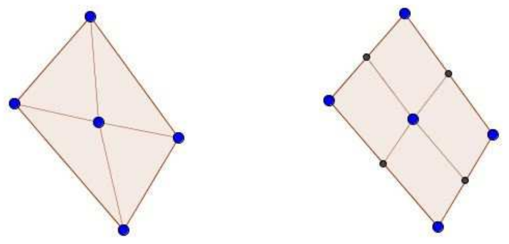


Figure 3.5: Step 1: Transformation of 4 triangles into 4 quads

Step 2: The second step unifies the biggest number of triangles in the mesh. For every triangle T, all the adjacent triangles T_i, if not already analyzed, are considered and the best quad, formed by T and T_j, is determined by analyzing the displacement G of the four quad corners with respect to a 90 degree angle.
In particular, for every couple of adjacent triangles T and T_i, the displacement G_i is computed by

$$G_i = \sum_{j=1}^{4} |(\alpha_j - 90°)| \tag{3.1}$$

where α_j are the angles of the quadrilateral obtained from T and T_i.
T is matched with the face T_i with the minimum G_i value.
Fig.3.6 shows the application of Step 2. Fig.3.6(a) illustrates the set of triangles $T_i,\ i = 1, 2, 3$ that can match with triangle T. According to the G_i values, the triangles T and T_1 are unified. The result of the match is shown in Fig.3.6(b).

Step 3. The main goal of this step is to determine if there is a different combination of triangles such that the number of non-matched triangles is minimized. In order to understand better this step we can consider the example in Fig.3.7. In Fig.3.7(a) we have two new quadruplet of quads, the green and the yellow one, bounded by non-matched triangles, respectively

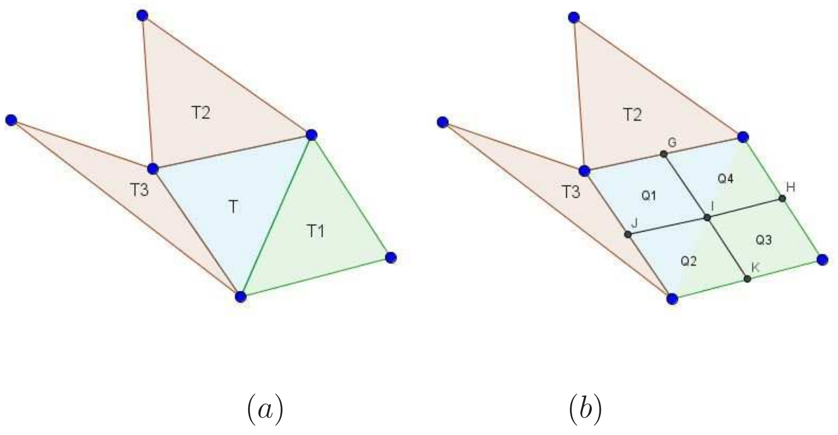


Figure 3.6: Step 2: Transformation of 2 triangles T and T_1 into 4 quads

the blue and the red ones.
If we apply the quadrilateral subdivision of remaining triangles we obtain the quads shown in Fig.3.7(b) and thus creating four new extraordinary vertices of valence 3. A more efficient quadrilateral subdivision can be obtained by matching the two quadruplets of triangles thus obtaining two sets of 8 quads without introducing any extraordinary vertex, as illustrated in Fig.3.7(c).

Step 4. The last step considers the remaining triangles and subdivides them into 3 quads introducing the middle point of every edge and the centroid of the triangle, as illustrated in Fig.3.8. In this case an extraordinary vertex of valence 3 is introduced.

The method previously described computes a quadrilateral mesh minimizing the number of triangles to be subdivided by the Step 4. This idea initially seems to be perfect because it minimizes the number of extraordinary vertices that are obtained during the decomposition. However we noticed that it is not enough, because there is no rule on the order in which triangles to be matched in couple are analyzed. This may produce a result in which two triangles A and B are matched because, analyzing A, B is the best matching for A. We improved this method by introducing two iterations in Step 2. In

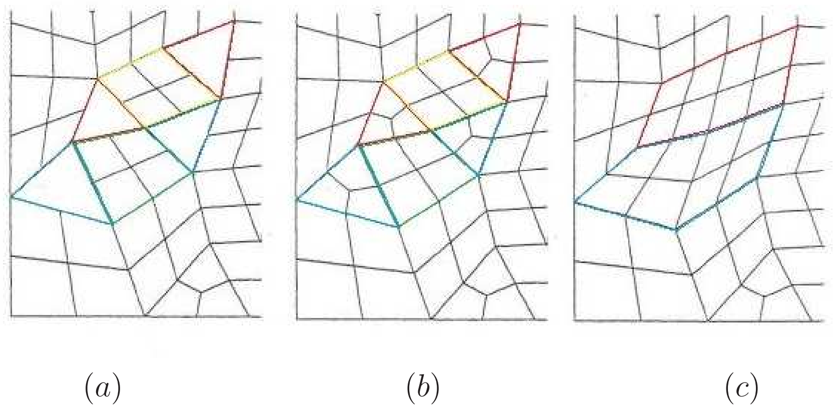


Figure 3.7: Step 3: Analysis of the remaining non-matched triangles

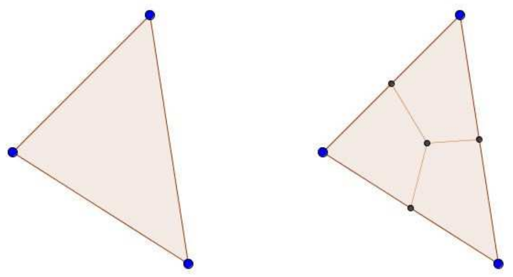


Figure 3.8: Step 4: Triangle subdivision into 3 quads

the first iteration, we match only couples of triangles such that $G_i < tol$, where G_i is defined in (3.1) and tol is a small angle (for example $tol = 10°$). This condition allows us to create quadrilaterals that are almost rectangles. Then, in the second iteration, we allow all the couple matching of the remaining triangles. In this way we obtain a better decomposition and the number of extraordinary vertices created by the subdivision is minimized.

After the conversion from triangular to quadrilateral mesh, we can optimize the global structure of the mesh in order to improve the quality of the quads in the quadrilateral mesh following for example the method described in [16].

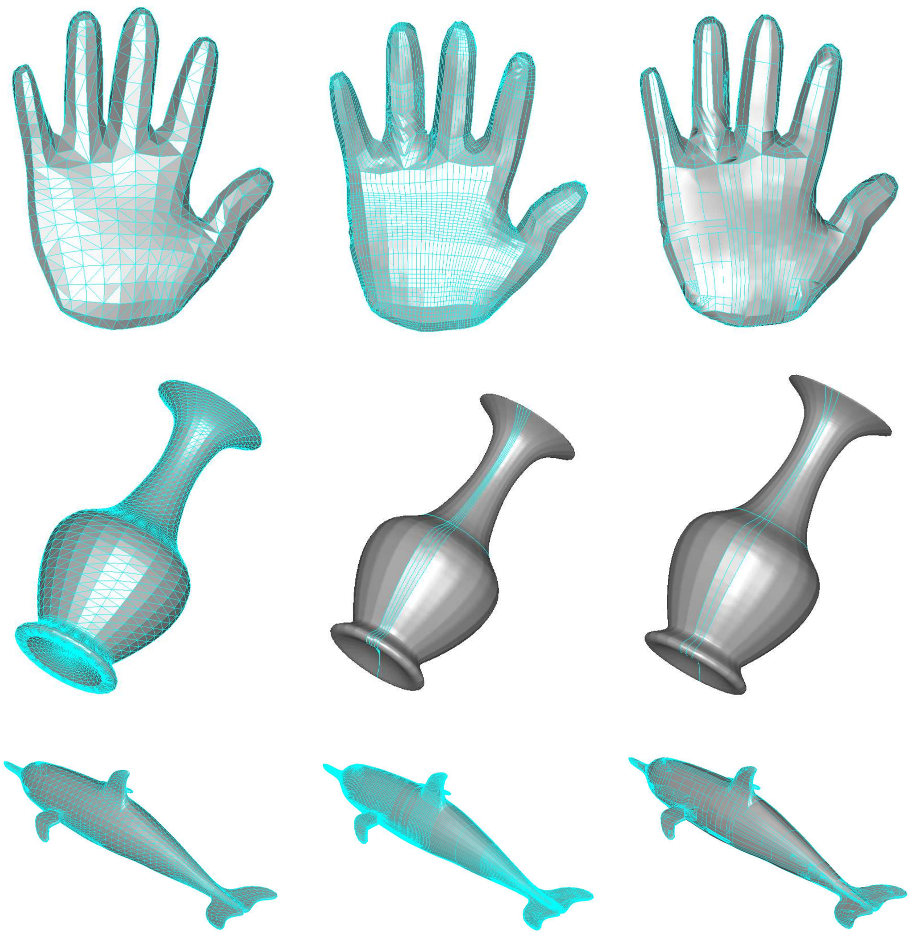


Figure 3.9: First Column: Triangular Meshes representing solid objects, Second Column: Extended B-Rep obtained by QMP, Third Column: Extended B-Rep obtained by QMTP. Faces are plotted with $3 \times 3gc$

Mesh	$\|F\|$	$\|V\|$	$\|EV\|$	$\|F_T\|$ QMP	$\|V_T\|$ QMP	$\|F_T\|$ QMTP	$\|V_T\|$ QMTP
Hand	1220	612	133	1534	1536	264	306
Vase	8832	4416	4	65	65	10	15
Dolphin	5360	2682	719	9480	9482	1453	2048

Table 3.3: EB-Rep form of Triangular Meshes (for meshes illustrated in Fig.3.9). From left to right: Mesh Name, number of faces, vertices, Extraordinary Vertices of M; number of faces in T by QMP, number of vertices in T by QMP, number of faces in T by QMTP, number of vertices in T by QMTP

Examples We tested the proposed method by implementing it in our OpenCascade platform. In Fig.3.9 we illustrate some examples of triangular meshes transformed into quadrilateral meshes and then efficiently represented by EB-Reps with bilinear NURBS surfaces using the QMP and QMTP algorithms presented. The results are reported in Table 3.3.
In this case the number of extraordinary vertices $|EV|$ is computed considering the quadrilateral mesh obtained as the result of the conversion. For triangular meshes we notice that only the QMTP algorithm gives an efficient representation of the triangular mesh with EB-Rep. The reason is that the tri-to-quad transformation increases the number of faces and sometimes produces an efficient representation without T junctions that has more faces than the original mesh.
These algorithms to obtain an efficient mesh representation by EB-Rep using NURBS surfaces solve problems connected to the representation of valence semi-regular meshes in standard B-Rep schemes. Moreover, if we want to compute a Boolean Operation between two Extended B-Rep solids we are forced to analyze geometrically both solid representations in order to find intersections between solids. In this situation, in order not to control all the plane faces of every mesh-face we could benefit of this efficient way to represent a mesh-face.
The general case of unstructured meshes is considered in chapter 4, where we introduce a new method to represent an unstructured triangular mesh as

an EB-Rep.

Chapter 4

EB-Rep form of an Unstructured Mesh

The faces of an EB-Rep can be mesh-faces, analytical surfaces and NURBS surfaces as desired. In our work we realized a geometric kernel, using the OpenCascade library, that extends a classic solid modeling system based on B-Rep in order to manage both meshes and NURBS faces.
In this chapter we face the problem to construct an EB-Rep with patching NURBS from an unstructured mesh representing the boundary of a solid object or part of it. This allows us to handle the mesh, described as an EB-Rep, in both a new Extended Solid modeling system and in our extension of a classic system. It is really important to observe that the EB-Rep allows us to manage separately the different parts of the solid. Some parts will be considered as Mesh-Faces and left unchanged, while others will be represented with NURBS patches and can be modified.
This problem arises for example in reverse engineering when a physical object is acquired by a 3D scanner system and reconstructed as unstructured mesh, subsequent CAD process that has to be performed on it will work on the associated B-Rep model.
In the literature there are a lot of methods to reconstruct a surface starting from an unstructured mesh or from a point cloud that is triangulated.

Most of the reconstruction methods used in reverse engineering subdivide the mesh into sub-meshes, then extract primitives that better fit the data and finally detect topology and boundaries of the object. In [11] a first approach that follows this pipeline is proposed for meshes and point clouds. An evolution of this approach is introduced in [36], where the authors propose a process to reconstruct a B-Rep model from a 3D point cloud. The first step consists of triangulating the point cloud. Then the authors propose to segment the mesh by using border edge detection and compute the primitive parameters for each sub-mesh with a method based on surface normal estimation. Topology is determined as in [11]. Recently in [20], the authors proposed a review of reverse engineering methods observing existent CAD systems.
Other methods analyze individually the extraction of primitives and the detection of topology and boundaries, often using RANSAC algorithm [30].
A recent method that realizes all the pipeline to reconstruct a B-Rep model composed of planes, spheres, cylinders and cones from a 3D mesh whose vertex coordinates are considered exact is introduced in [14]. The first seminal proposal for patching NURBS are [32] and [35], where Hoppe proposed a method for Unorganized Points. This method is a surface reconstruction method that triangulates the point cloud by means of energy minimization producing an unstructured triangular mesh. The complexity of this method is directly related to the arbitrary topology of the object represented.

Our method creates an EB-Rep with NURBS faces approximating an unstructured triangular mesh with arbitrary topology. It can be applied to obtain a valence semi-regular mesh from an unstructured one.
The method is subdivided in the following steps:

. Step 1: create a simplified quadrilateral mesh M_q by approximating the original unstructured mesh M

. Step 2: compute a parametrization for every set of points

. Step 3: apply LSPIA algorithm adapted to approximate points in order to create a Catmull-Clark surface approximating the original points.

. Step 4: convert the Catmull-Clark surface to NURBS patches.

In the next sections we describe in more details the required steps.

4.1 Step 1: From a Triangular Mesh to a simplified quad-mesh

Given a triangular unstructured mesh M, the method described in [38] is applied to obtain a quad dominant simplified mesh. The algorithm, called Instant Field-aligned Mesh, computes a mesh M_{qd} that is globally aligned with a direction field using local orientation-field and local position-field smoothing operators. The mesh is then extracted from the fields and optionally post-processed. The number of faces of M_{qd} is decreased significantly compared to the initial number of triangle faces in M. The topology of the surface is preserved.
Fig.4.1 shows an example of simplification of a mesh representing a section of an Artery obtained using a 3D scanner. Fig.4.1(a) shows the original vertices of the mesh, Fig.4.1(b) illustrates the triangulation obtained from the vertices. The simplified mesh is shown in Fig4.1(c) and, overimposed, the vertices of M (Fig4.1(d)).
By observing the obtained mesh we notice that the remeshing algorithm produces a few triangular and pentagonal faces.

The simplified quad dominant mesh M_{qd} is transformed into a quadrilateral mesh M_q according to the following rules necessary to apply the step 2:

. If a face has an extraordinary vertex, it has to be stored as the first vertex of the face.

. Quad faces obtained from a non-quadrilateral face have to be stored in counterclockwise order starting from the first vertex.

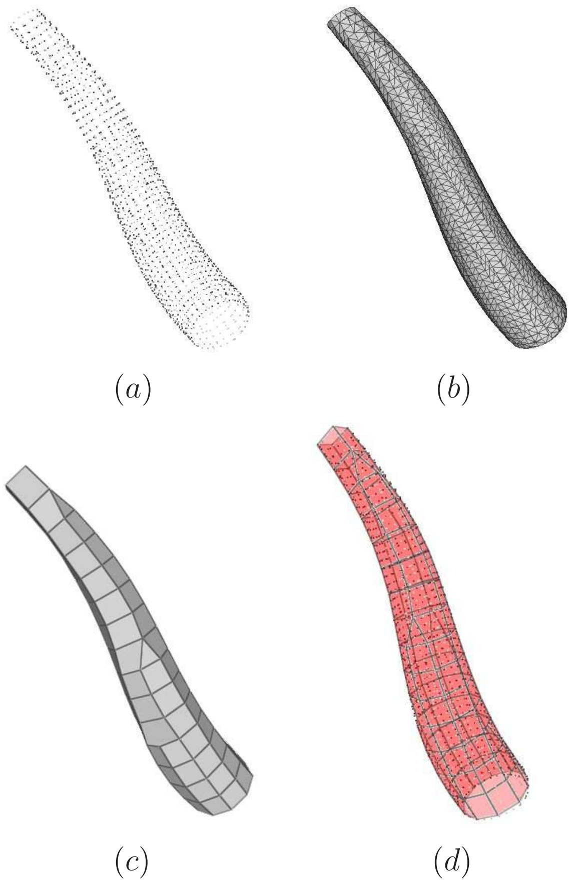


Figure 4.1: Artery Point Cloud and Artery Triangulation

In order to satisfy these conditions, we transform every non-quadrilateral face F_i, subdividing it into quads. Generally, for a polygonal face F_i with n edges, the centroid c_i of F_i is connected to the middle points $m_{i1}, m_{i2}, \ldots, m_{in}$ of every edge. Therefore n quadrilateral faces are obtained. Fig.4.2 shows how both pentagonal and triangular faces are transformed into quadrilateral faces.
Result is a non-conformal mesh that is used to associate a quadrilateral face with every point. Step 3, which performs the LSPIA algorithm, will receive as input M_{qd} and other informations about connection between M_{qd}, M_q and the vertices of M.
In the last part of Step 1, the order of the vertices in every face is modified

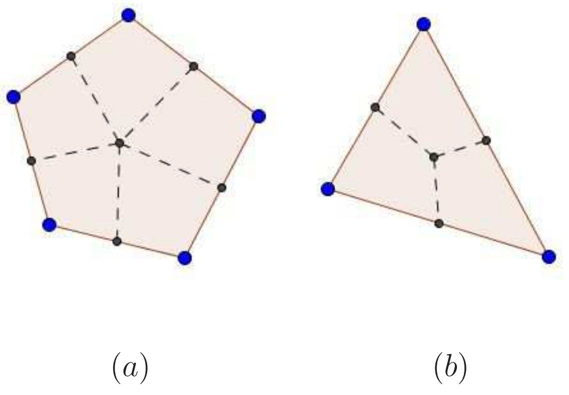


Figure 4.2: Examples of subdivision of a polygonal face in quads: a) pentagonal face, b) triangular face

keeping orientation and ordering vertices such that the first vertex of the face is an extraordinary vertex, if it exists.
Fig.4.3 shows an example of the transformation of the mesh M into the quadrilateral mesh M_{qd}. As we can notice we obtain a non-conformal mesh in which new edges, created subdividing triangular and pentagonal faces, delimitate a quadrilateral face.

This new non-conformal structure allows us to keep the original quad domi-

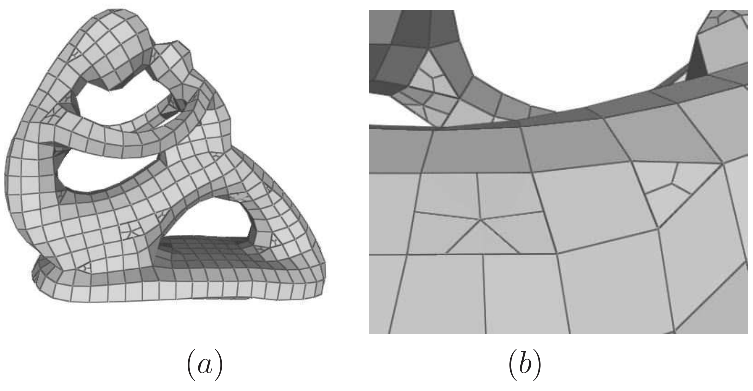


Figure 4.3: Fertility mesh with quadrilateral faces: a) M_q mesh, b) a zoomed detail

nant mesh and to know exactly which new quadrilateral faces are associated with an old non-quadrilateral face. This does not create any problem for the association of points with a face, because in this case only the new quadrilateral faces are considered.

4.2 Step 2: Parametrization

The created coarse quadrilateral mesh M_q is a non-conformal quadrilateral mesh approximating M. We associate with every face $\check{f}_i$ of M_q a cloud of points $P_{\check{f}_i}$, subset of the vertices of M. This is performed associating with every point the nearest face intersected by its normal vector.
For every face $\check{f}_i$ with internal points $P_{\check{f}_i}$ and vertices $v_1, \ldots, v_4$ the parametrization associates the vertices with the corners of the planar domain $[0,1]^2$, such that vertices have parametric coordinates (u,v) respectively $(0,0)$, $(0,1)$, $(1,1)$, $(1,0)$ and internal points have coordinates (u,v) with $0 \leq u,v \leq 1$. In Fig.4.4 an example of association between points and faces is illustrated.

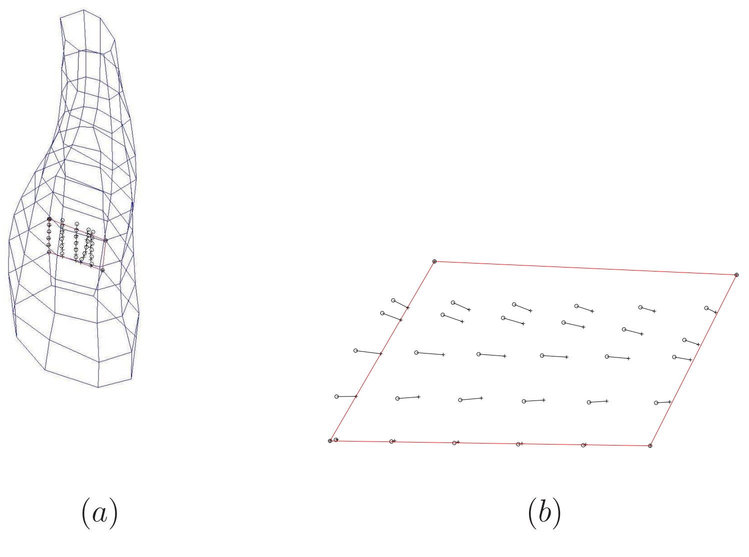

(a) (b)

Figure 4.4: Points associated with a face of *Artery*

Fig.4.4(a) shows the coarse mesh M_q and a group of points associated with a given face. Fig.4.4(b) illustrates a detail of the given face and all points associated with it, with a line connecting every point with the corresponding point on the face.

4.3 Step 3: Application of LSPIA Approximating Algorithm

In [42] a Progressive Iterative Approximation algorithm (PIA) for curves and Loop subdivision surfaces is introduced with the aim of constructing efficiently smooth interpolations of a set of points. Then, in [22] the PIA proposal is extended for interpolating a set of points with a Catmull-Clark surface. In [39] a PIA algorithm is introduced in order to interpolate/approximate a B-Spline surface with rectangular topology. Finally, in [25] the PIA algorithm is extended to a Least Squares Approximating algorithm (LSPIA) for NURBS surfaces.
We extended LSPIA algorithm to obtain a Catmull-Clark surface [19] approximating with good accuracy the original mesh M.
Given a mesh M, a quad dominant mesh M_{qd} extracted by M, with $n_v + 1$ vertices and $n_f + 1$ faces, and the associated quadrilateral coarse mesh M_q, with $m_v + 1$ vertices and $m_f + 1$ faces. For every point Q_i in the set V of vertices of M, we have a corresponding face $\check{f}_j$ $j = 0, \ldots, m_f$ in M_q and f_l $l = 0, \ldots, n_f$ in M_{qd} and parametric coordinates (u_i, v_i) associated with Q_i on $\check{f}_j$. Fig.4.5 shows the difference between faces of M_{qd} (on the left) and M_q (on the right). In particular $\check{f}_j$ is a quadrilateral face of M_q obtained subdividing a pentagonal polygon, while f_l is a face of M_{qd}. We notice that each face $\check{f}_j$ of M_q has a corresponding face f_l of M_{qd}. LSPIA iteratively constructs the approximating surface as a Catmull-Clark Surface. Let be given an ordered point set $\{Q_i\}_{i=0}^{m_v}$ to be fitted and $\{(u_i, v_i)\}_{i=0}^{m_v}$ the associated parameters, with $(u_i, v_i) \in \Omega_j \quad j = 0, \ldots, m_f$; Ω_j is the parametric domain $[0, 1]^2$ associated with $\check{f}_j$ in M_q. At the starting iteration we define

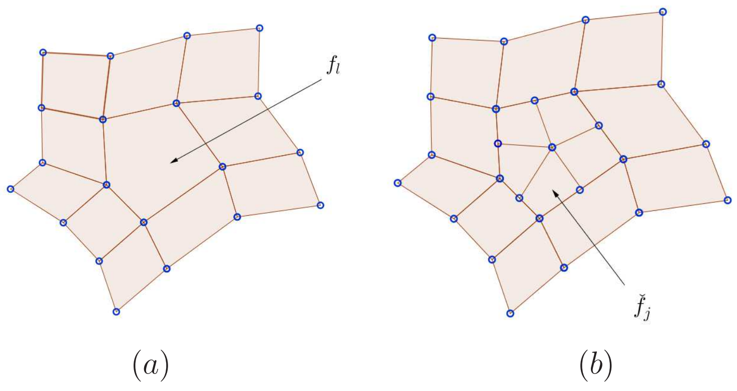


Figure 4.5: a) Faces of M_{qd} b) Faces of M_q

$\{P_h^0\}_{h=0}^{n_v}$ that are the vertices of M_{qd} as the control points of the blending Catmull-Clark surface $\mathbf{P}^0$ i.e.,

$$\mathbf{P}^0(u,v) = \sum_{h=0}^{n_v} B_h(u,v) P_h^0 \qquad (u,v) \in \Omega \quad \Omega = \cup_{i=0}^{m_f} \Omega_j$$

where $B_h(u,v)$ are the blending basis functions of the Catmull-Clark surface in a space of dimension $n_v + 1$. Then the displacement with respect to the original points is computed as follows

$$\delta_i^0 = Q_i - \mathbf{P}^0(u_i, v_i) \qquad i = 0, \dots, m_v. \tag{4.1}$$

The adjusting vector is defined as

$$\Delta_h^0 = \mu \sum_{i=0}^{m_v} B_h(u_i, v_i) \delta_i^0 \qquad h = 0, \dots, n_v \tag{4.2}$$

where μ is a constant satisfying the condition

$$0 < \mu < \frac{2}{\lambda_0} \tag{4.3}$$

with λ_0 the largest eigenvalue of matrix $A^T A$, where A, is the collocation matrix defined as:

$$A = \begin{pmatrix} B_0(u_0, v_0) & B_1(u_0, v_0) & \dots & B_n(u_0, v_0) \\ B_0(u_1, v_1) & B_1(u_1, v_1) & \dots & B_n(u_1, v_1) \\ \vdots & \vdots & \ddots & \vdots \\ B_0(u_m, v_m) & B_1(u_m, v_m) & \dots & B_n(u_m, v_m) \end{pmatrix}$$

The new control points at the next iteration are

$$P_h^1 = P_h^0 + \Delta_h^0 \qquad h = 0, \dots, n_v \tag{4.4}$$

and the new fitting surface is defined as

$$\mathbf{P}^1(u, v) = \sum_{h=0}^{n_v} B_h(u, v) P_h^1 \qquad (u, v) \ \in \ \Omega \tag{4.5}$$

In general, at the k-th iteration, starting from the surface $\mathbf{P}^k(u, v)$, we compute

$$\delta_i^k = Q_i - \mathbf{P}^k(u_i, v_i) \qquad i = 0, \dots, m_v,$$

$$\Delta_h^k = \mu \sum_{i=0}^{m_v} B_h(u_i, v_i) \delta_i^k \qquad h = 0, \dots, n_v,$$

$$P_h^{k+1} = P_h^k + \Delta_h^k \qquad h = 0, \dots, n_v.$$

Then the surface at the $(k+1)$-th iteration

$$\mathbf{P}^{k+1}(u, v) = \sum_{h=0}^{n_v} B_h(u, v) P_h^{k+1} \qquad (u, v) \ \in \ \Omega$$

In our algorithm, at the first step, it is necessary to apply Stam's algorithm [48] in order to evaluate the basis functions in the matrix A.
Stam's algorithm requires one or two steps of refinement of the mesh M_{qd}, that can be either local or global. In our implementation we used a local refinement that significantly improves performances and saves memory space.

This allows us to define Catmull-Clark surfaces on Ω_j associated with $\breve{f}_j$ in M_q as follows:

$$S(u,v)_{|(u,v)\in\Omega_j} = \sum_{i=1}^{K} \breve{P}_h^0 b_i(u,v) \tag{4.6}$$

where $K = 2N + 8$ and N is the valence of the EV of $\breve{f}_j$. If $N = 4$, $\breve{P}_h^0$ are exactly the vertices P_h^0 of M_{qd}, otherwise they are a first or a second local Catmull-Clark refinement of the vertices P_h^0. Fig.4.6 shows a local first refinement of M_{qd} to obtain the vertices $\breve{P}_h^0$. Vertices in blue are the $2N+8$ vertices necessary to evaluate the surface in Ω_j corresponding to the face in blue. In this case $N = 5$, thus we have 18 vertices. We compute every

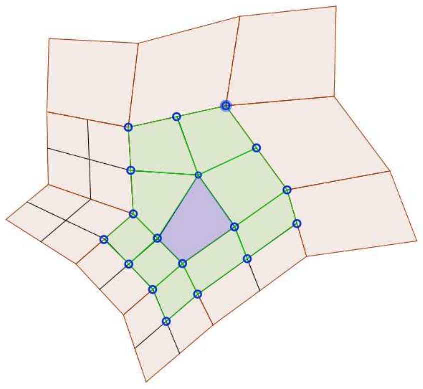


Figure 4.6: First refinement of a face with an extraordinary vertex

basis function $B_h(u,v)$ by using (4.6). In particular for each $h = 0,\ldots,n_v$ we define

$$q_j = \begin{cases} 0 \text{ if } j \neq h \\ 1 \text{ if } j = h \end{cases}$$

and

$$B_h(u,v) = \sum_{i=1}^{K} \breve{q}_i b_i(u,v) \quad (u,v) \in \Omega$$

where $\breve{q}_i$ are a first or a second local Catmull-Clark refinement of the scalar coefficients q_j.

To avoid the computation of the eigenvalues of $A^T A$, in [25] an alternative method to determine the weight μ is proposed. A is a $(m+1)\times(n+1)$ matrix. Let $A^TA = \{a_{i,j}\}_{0,0}^{m,n}$ where $a_{i,j} = \sum_{k=0}^{m} B_i(u_k,v_k)B_j(u_k,v_k)$.

Together with $\sum_{j=0}^{n} B_i(u_k,v_k) = 1$ we have

$$\begin{aligned}\sum_{j=0}^{n} a_{i,j} &= \sum_{j=0}^{n}\left[\sum_{k=0}^{m} B_i(u_k,v_k)B_j(u_k,v_k)\right]\\ &= \sum_{k=0}^{m} B_i(u_k,v_k)\left[\sum_{j=0}^{n} B_j(u_k,v_k)\right] = \sum_{k=0}^{m} B_i(u_k,v_k) =: c_i\end{aligned}$$

It means that c_i is the sum of the i-th row elements of A^TA.
Therefore, $\lambda_0 \le \max_i\ c_i := C,\ i=0,\ldots,n$ and $\frac{2}{C} < \frac{2}{\lambda_0}$, so we define

$$\mu = \frac{2}{C}.$$

Theorem 4.3.1. *If A is non-singular, the introduced LSPIA method is convergent.*

Proof. As result of the iterative method introduced above, a sequence $P^k(u,v)$ $k=0,1,\ldots$ is generated. To show its convergence, let

$$P^k = \{P_0^k, P_1^k, \ldots, P_n^k,\}^T$$

and

$$Q = \{Q_0, P_1, \ldots, P_m,\}^T$$

According to (4.4) we have

$$\begin{aligned}P_i^{k+1} &= P_i^k + \mu\sum_{j=0}^{m} B_i(u_j,v_j)(Q_j - P^k(u_j,v_j))\\ &= P_i^k + \mu\sum_{j=0}^{m} B_i(u_j,v_j)\left[Q_j - \sum_{l=0}^{n} B_l(u_j,v_j)P_l^k\right]\end{aligned}$$

then, we get,

$$P^{k+1} = P^k + \mu A^T(Q - AP^k) \tag{4.7}$$

where A is the collocation matrix.
Letting I be the $n+1$ identity matrix and $D = I - \mu A^T A$, by (4.7) we have

$$\begin{aligned} P^{k+1} - (A^TA)^{-1}A^TQ &= (I - \mu A^TA)[P^k - (A^TA)^{-1}A^TQ] \\ &= (I - \mu A^TA)^2[P^{k-1} - (A^TA)^{-1}A^TQ] \\ &= \dots \\ &= D^{k+1}[P^0 - (A^TA)^{-1}A^TQ] \end{aligned}$$

Supposing $\{\lambda_i(D)\}_{i=0}^n$ are the eigenvalues of D sorted in non-decreasing order, we get $\lambda_i(D) = 1 - \mu\lambda_i$ where $\{\lambda_i\}_{i=0}^n$ are the eigenvalues of A^TA sorted in non-decreasing order.
We supposed that A is non-singular and A^TA is positive definite so, noting $0 < \mu < \dfrac{2}{\lambda_0}$, we have $0 < \mu\lambda_i < 2$ and $-1 < \{\lambda_i(D)\} < 1 (i = 0, 1, \dots, n)$. It leads to $0 < \rho(D) < 1$, where $\rho(D)$ is the spectral radius of D, and the convergence condition is satisfied. $\square$

If A is singular, A^TA is not positive definite and we do not have a theoretical proof of the convergence. However it should be pointed out that we have made lots of experiments with this method and all of them converged also in the case of singular collocation matrix.
Given a tolerance tol and defined $dist(k) = \|P^k(u_i, v_i) - Q_i\|_2$, the iterative procedure is stopped when one of the following conditions is met:

$$|dist(k+1) - dist(k)| < tol$$

$$dist(k+1) < tol.$$

In our implementation we used $tol = 1 \times 10^{-4}$. A Catmull-Clark surface is obtained as result.

4.4 Step 4: Patching NURBS

As known in literature, Catmull-Clark surfaces have problems of continuity and curvature around the EV. In order to solve this problem, many patching techniques have been proposed. A first approach for the correction of the surface consists in blending the surface in order to obtain the desired regularity. A second approach replaces the Catmull-Clark surface around EV with a patch that preserves continuity and improves curvature. Other different solutions for this problem are proposed and cited in [9].
In our work we use a Gregory patch to replace Catmull-Clark patch [9]. In particular, a Catmull-Clark patch with an extraordinary point is replaced by a bicubic Gregory patch that interpolates the boundary of the patch itself and joins with G^1 continuity with the adjacent patches. This process is explained in details in [9] and involves the computation of first derivatives of the vertices. The bicubic patch is created by modifying a classic bicubic

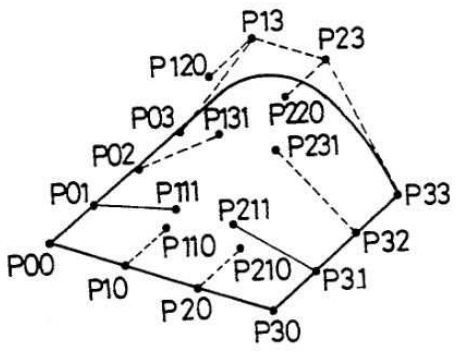


Figure 4.7: Control Points in a Bicubic Gregory patch

Bézier patch. In particular, as shown in Fig.4.7, the four constant internal control points are modified by inserting four control points that are a convex combination of 2 assigned points obtained by imposing G^1-continuity conditions at the boundaries. In this way the bicubic patch is defined by 20 control points, while the classic bicubic Bézier patch is defined by 16 control points. The internal control points are $P_{1,1,i}$, $P_{1,2,i}$, $P_{2,1,i}$, $P_{2,2,i}$ with $i = 0, 1$.

All adjacent patches around the EV are created in a similar way in order to have a G^1 connection.
In particular a bicubic Gregory patch has the following formula:

$$S(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_i^3(u)B_j^3(v)P_{i,j}(u,v) \tag{4.8}$$

where:

$$B_i^n(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$$

is the usual basis of the Bernstein polynomials and the internal control points are given by

. $P_{1,1} = \dfrac{uP_{1,1,0} + vP_{1,1,1}}{u+v}$

. $P_{1,2} = \dfrac{uP_{1,2,0} + (1-v)P_{1,3,1}}{u+(1-v)}$

. $P_{2,1} = \dfrac{(1-u)P_{2,1,0} + vP_{2,1,1}}{(1-u)+v}$

. $P_{2,2} = \dfrac{(1-u)P_{2,2,0} + (1-v)P_{2,3,1}}{(1-u)+(1-v)}$

Obtained result is a patching in which all regular boundaries have a C^2 connection, while patches that share an extraordinary vertex are G^1 joined.
Finally, a Gregory patch can be represented as a rational NURBS patch [49]. The patch (4.8) can be described using a rational Bézier patch of degree 7 in u and v.
To sum up, the steps described above allow us to obtain a NURBS representation of a Catmull-Clark surface approximating an unstructured polygonal mesh.

4.5 Examples

We report the results of our algorithm when applied to two differents unstructured meshes “Artery” and “Fertility”. The first mesh represent an open section of an artery, while the second one a closed mesh of a statue.

Fig.4.8 shows the algorithm applied to the mesh that represents a section of an artery. Fig.4.8(a) shows the set of vertices V and the open quad-dominant mesh M_{qd} created by Instant Field-aligned algorithm.

Fig.4.8(b) shows a zoom of Fig.4.8(a) in which it is possible to observe vectors connecting vertices and the corresponding points on the associated face. Fig.4.8(c) shows the set of vertices V and the Catmull-Clark result surface grid, that allows us to see all the extraordinary vertices of the approximating surface. Finally, Fig.4.8(d) shows the NURBS patches in which the Catmull-Clark surface is subdivided in order to obtain an approximating EB-Rep with NURBS surfaces.

In order to obtain this result, LSPIA algorithm has been applied on a mesh M_q with 127 faces, setting the value μ defined in (4.3), $\mu_{artery} = 9.8 \times 10^{-2}$. Result has been obtained in 24 iterations, with a residual tolerance of 1×10^{-4} and approximates the original mesh with a residual $r = 1.462 \times 10^{-1}$.

Fig.4.9 shows the algorithm applied to the mesh that represents the Fertility statue. Fig.4.9(a) shows the set of vertices V and the open quad-dominant mesh M_q created by Instant Field-aligned. Fig.4.9(b) shows a zoom of Fig.4.9(a) in which it is possible to observe vectors connecting vertices and the corresponding points on the associated face.

Fig.4.9(c) shows the vertices V and the Catmull-Clark resulting patching, that allows us to see all the extraordinary vertices of the approximating surface. In Fig.4.9(d) the NURBS patches in which the Catmull-Clark surface is subdivided in order to obtain an approximating B-Rep with NURBS surfaces are illustrated. In order to obtain this result, LSPIA algorithm has been applied on a mesh M_q with 1090 faces, setting the value μ defined in 4.3, $\mu_{artery} = 2.14 \times 10^{-2}$.

Result has been obtained in 53 iterations, with a residual tolerance of 1×10^{-4} and approximates the original mesh with a residual $r = 2.68 \times 10^{-1}$.

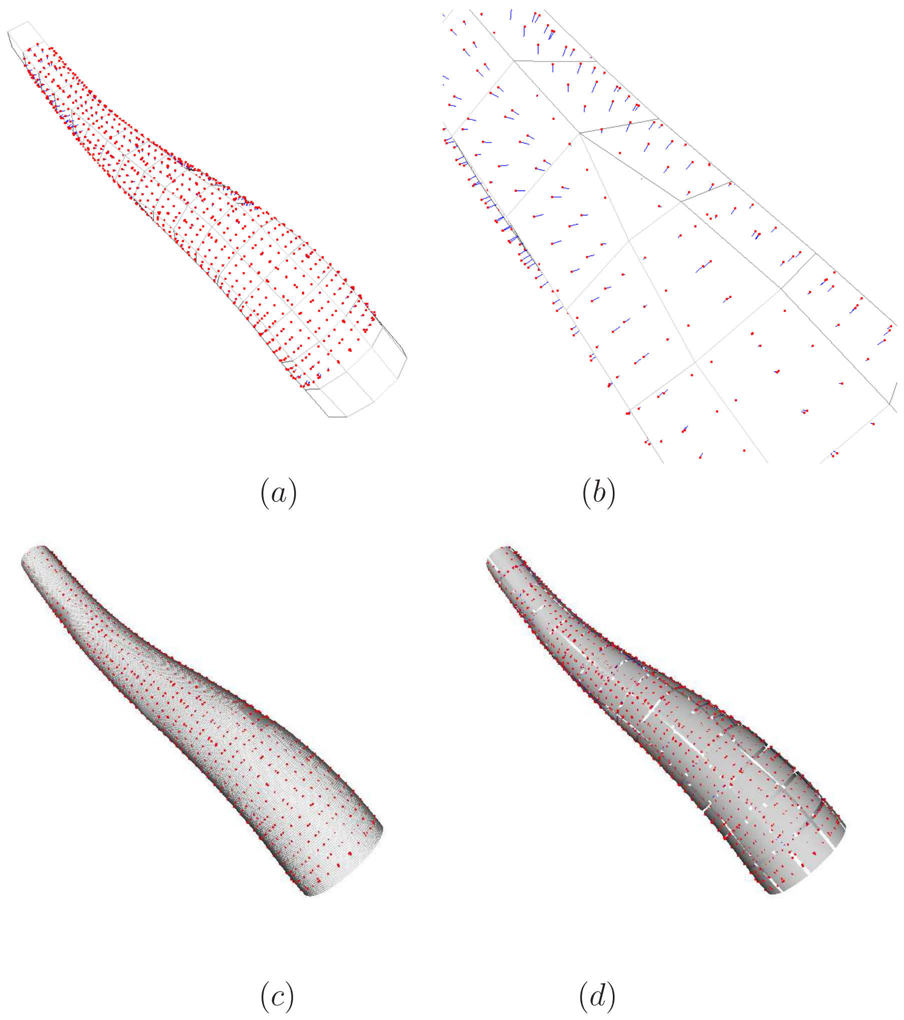


Figure 4.8: Artery: a) Vertices and Coarse Quadrilateral Mesh, b) Zoom that shows vectors connecting Vertices and corresponding points on associated faces, c) Vertices and Catmull-Clark result surface grid, d) NURBS patches partially visualized in order to show the shape and the position

4.6 A proposal of a Surface Reconstruction Method

Surface reconstruction methods create a 3D model from a dense cloud of points representing a solid object. In the literature there are a lot of methods to reconstruct a surface starting from a point cloud. Several approaches are based on combinatorial structures, such as Delaunay triangulations [40], alpha shapes [13], or Voronoi diagrams [8]. These schemes create a triangle mesh that interpolates all or most of the points of the original point cloud. Other methods directly reconstruct an approximating surface, typically represented in implicit form. These methods can be subdivided in global and local methods. Global methods commonly define the implicit function as the sum of radial basis functions (RBF) [18]. Local methods consider subsets of nearby points at a time.
According to a recent state of the art report [12], almost all surface reconstruction methods produce as output an implicit representation or a triangular mesh. One of the few methods that creates NURBS patching is the Hoppe's method for Unorganized Points, introduced in [32] and in [35]. Another interesting method that fits a triangular mesh with G^1 smoothly stitching bi-quintic Bézier patches is introduced in [41].
Our method can be considered as a variant of Hoppe's surface reconstruction method which works on an unstructured mesh obtained by a pre-processing of the original point cloud. In Hoppe's work a triangular mesh is obtained from a point cloud applying a method that defines an implicit signed distance function associating an oriented plane with each of the points. Then the mesh is obtained applying a contouring algorithm. Finally an optimization algorithm is applied in order to reduce the number of vertices and to fit well the point cloud. Details of this method are described in [34] and in [33].
Another possibility is to apply the Cocone algorithm that reconstructs a mesh from sample points. This method was introduced in [7]. The input consists of the coordinates of the point cloud and output is a piecewise linear approximation of the surface which is made of Delaunay triangles with vertices at the input points only. Other versions of the algorithm have been

realized in order to manage noisy point clouds or large sets of points.

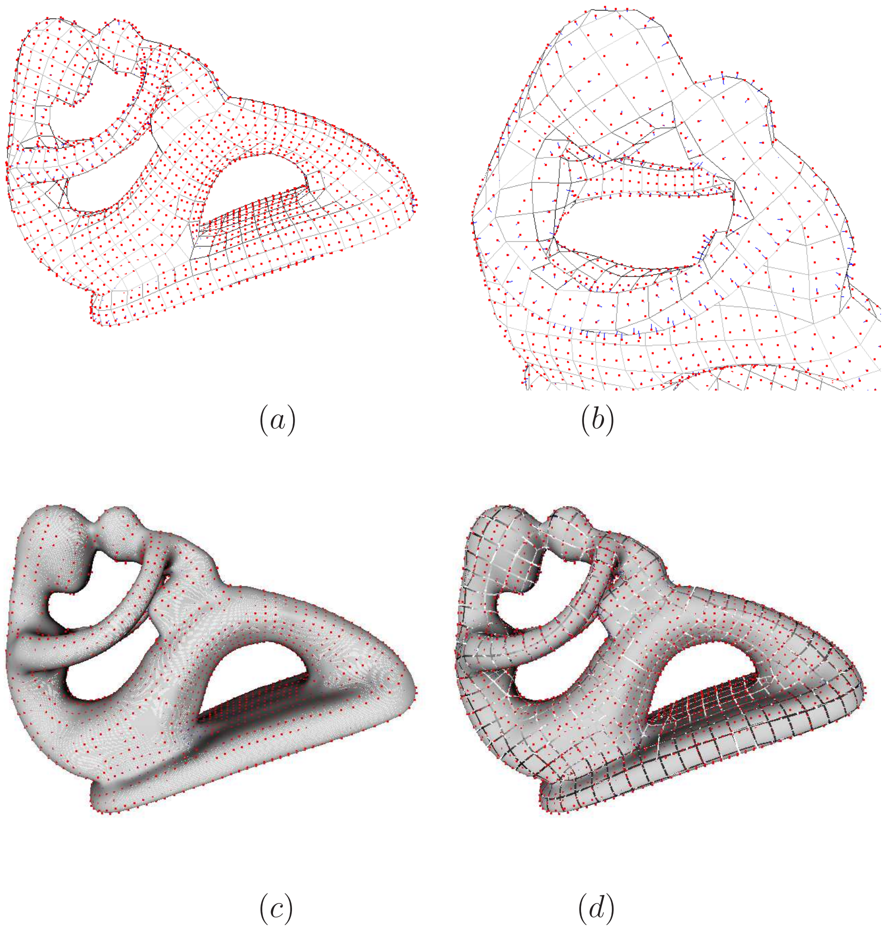


Figure 4.9: Fertility Statue: a) Vertices and Coarse Quadrilateral Mesh, b) Zoom that shows vectors connecting Vertices and corresponding points on associated faces, c) Vertices and Catmull-Clark result surface grid, d) NURBS patches partially visualized in order to show the shape and the position

Chapter 5

Solid Modeling tools for Extended B-rep

We introduced Extended B-Rep schemes and described some methods to provide an efficient representation of both valence semi-regular and unstructured meshes as EB-Reps. We want now to extend the most important tools for solid modeling to manipulate EB-Rep solids. Although examples illustrated in this chapter are realized using our geometric kernel, that is an extension of a classic system based on B-Rep data structure, they allow us to understand how these tools could be applied to EB-Rep solids in a new Extended Solid Modeling System.
In the first part of this chapter, we describe the principal tools of solid modeling: Boolean Operations and Cutting Operation. In the second part we analyze the Join between EB-Rep models, providing some basic notions and examples.

Generally the construction of solid objects requires a long sequence of simple editing operations which are the fundamental tools for Solid Modeling. The following tools have been considered and realized in our Extended Solid Modeling System:

. Affine transformations: rotation, scaling and translation

. Boolean Operations

. Cutting Operation

. Join

Affine transformations, such as rotation, scaling and translation, are immediatly applied to the EB-Rep schemes because the topological structure of the EB-Rep solid is unchanged and the geometry entities associated follow the same rules used for standard B-Rep solids [43]. The other operations are considered in the rest of this chapter.

5.1 Boolean Operations between EB-Reps

In chapter 2 we overviewed Boolean Operations, that are the most important tools used to model a solid object. The combination of Union, Intersection and Difference operations on simple shapes allow us to create complex objects.
In order to compute Boolean Operations between EB-Reps the same rules used for standard Boolean Operations between B-Reps can be applied. However, as explained in chapter 2, the introduction of the Mesh-Face primitive implies new difficulties both in performing the intersection between a Mesh-Face and a NURBS or analytical face and in intersecting Mesh-Faces.
We realized Boolean Operations between EB-Rep solids in the Extended Solid Modeling System. We report a few examples of Boolean Operations between a solid represented by NURBS surfaces and a solid described by Mesh-Faces. Fig.5.1(a-b) show a B-Rep cylinder built with NURBS surfaces and a Mesh-Face representing a tooth. Fig.5.1(c) shows the EB-Rep result of the Boolean Union between the cylinder and the tooth which is composed of NURBS surfaces and Mesh-Faces. Observing the result, we notice that both NURBS surfaces and Mesh-Faces are trimmed. The intersection curves

are trimmed NURBS curves, if they delimit two NURBS surfaces; otherwise, they are polylines that have AG^0 continuity with both the NURBS surface and the Mesh-Face.
In this example the new EB-Rep scheme $B_e = (G_e, T)$ has the geometric set G_e made of points, curves, surfaces and meshes. Here curves are trimmed NURBS and polylines involving points on both NURBS surfaces and Mesh-Faces, surfaces are trimmed NURBS, while the set of meshes contains trimmed Mesh-Faces. A further example is illustrated in Fig.5.2, where

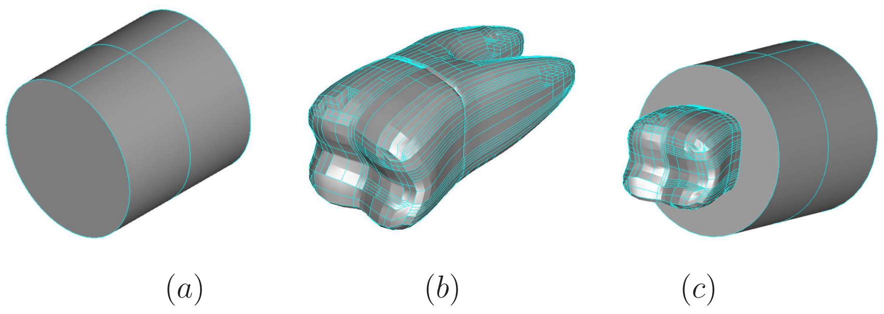


Figure 5.1: Example of Boolean operation: a) B-Rep (NURBS) cylinder, b) EB-Rep (Mesh-Face) representing a tooth. c) Union result. Faces are plotted with $3 \times 3gc$

a NURBS cylinder, a Mesh-Face representing a tooth and a NURBS cube are used to create a prosthesis for a dental implant. Fig.5.2(a) shows the three objects. Fig.5.2(b) shows the result of the Boolean Difference between the tooth and the cube. In this case the new EB-Rep obtained is bounded by a trimmed Mesh-Face and a trimmed NURBS representing a plane. The intersection curve is a closed polyline that has AG^0 continuity with both the NURBS and the Mesh-Face surfaces.
Fig.5.2(c) shows the result of the difference between EB-Rep obtained in Fig.5.2(b) and a NURBS cylinder. The cylindric face in the result is a trimmed NURBS, while the intersection curve is a closed NURBS curve. Boundary curves are two trimmed NURBS and a polyline involving points on the NURBS plane surface and the Mesh-Face. The set of surfaces is made

of three trimmed NURBS, while the set of meshes contains a trimmed Mesh-Face.

The last and more general example is illustrated in Fig.5.3, where the lower

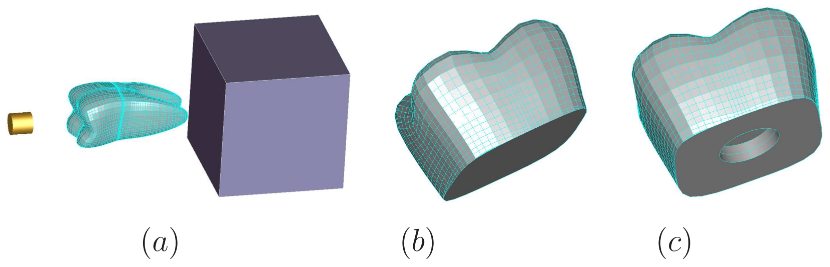


Figure 5.2: Example of Boolean operation: a) a B-Rep (NURBS) cylinder, an EB-Rep (Mesh-Face) tooth and a NURBS cube, b) difference between tooth and cube. c) difference between result of operation in b) and cylinder. Faces are plotted with $3 \times 3gc$

part of a shoe is created using boolean operations between EB-Reps with both Mesh-Faces and NURBS surfaces. In particular Fig.5.3(a-b-c) show respectively a B-Rep solid, bounded by NURBS surfaces, representing the "Air Jordan logo", a Mesh-Face representing the lower part of a sole and a B-Rep solid made of NURBS surfaces that represents the insole of the shoe. Fig.5.3(d-e) show the result of Boolean Union between the sole and the insole. In this case the new EB-Rep obtained is bounded by both a trimmed Mesh-Face and a trimmed NURBS. The intersection curve is a closed polyline that has AG^0 continuity with both the NURBS and the Mesh-Face surfaces. Fig.5.3(f-g) show the result of Boolean Difference between the EB-Rep obtained at the previous step and the "Air Jordan logo". As we can notice observing the detail in Fig.5.3(g), in the new EB-Rep the Mesh-Face and the NURBS solid are trimmed and the intersection polyline has AG^0 continuity with both the NURBS and the Mesh-Face surfaces.

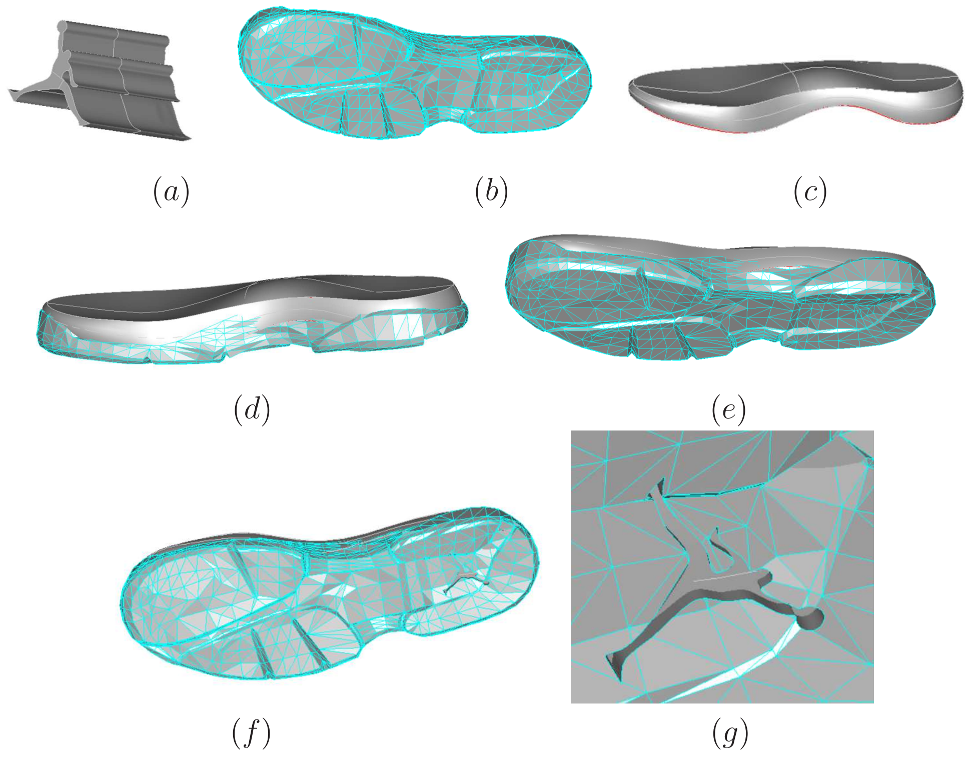


Figure 5.3: Example of Boolean operation: a) a B-Rep (NURBS) solid representing the “Air Jordan logo”, b) an EB-Rep (Mesh-Face) representing a sole, c) a NURBS insole, d-e) union between insole and sole, f) difference between result of operation in d) and “Air Jordan logo”, g) Detail of boolean operation in f).

5.2 Cutting Operation

In chapter 2 we introduced Cutting operation as a fundamental tool of “Hybrid Solid Modeling Systems”. In particular cutting involves a solid A that is cut by a surface s. Result of the operation are two distinct solid components that share the common face s. According to the outward normal of the surface, one of the two solids is chosen. We extended cutting operation to allow also EB-Rep solids and Mesh-Faces as input. An EB-Rep solid can be cut both by a NURBS face and a Mesh-Face. Result is always an EB-Rep solid. Fig.5.4 shows an example of Cutting with Extended B-Rep models. In

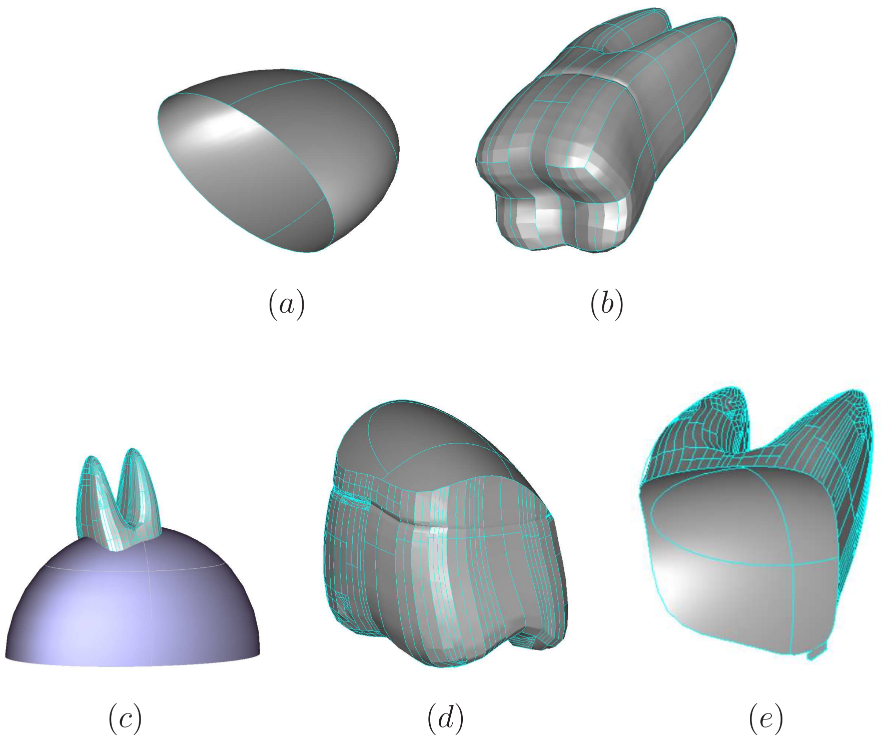


Figure 5.4: Example of Cutting operations with Extended B-Rep. First row: a)NURBS surface, b) Mesh-Face representing a tooth. Second row: a) surface and Mesh-Face b) inferior solid obtained by cutting c) superior solid obtained by cutting. Faces are plotted with $3 \times 3gc$

particular, Fig.5.4(a-b) show the cutting surface (a) and the solid to be cut (b). The solid is an EB-Rep description of a mesh representing a tooth, while the surface is a NURBS surface. We placed the two objects as displayed in Fig.5.4(c) such that the surface subdivide the solid into two separated components. The resulting solids shown in Fig.5.4(d) and Fig.5.4(e) are two Extended B-Rep with NURBS faces and mesh-Faces. As we can notice, both the EB-Rep schemes $B_e = (G_e, T)$ associated with the two resulting solids contain a trimmed Mesh-Face, a trimmed NURBS surface and a polyline having AG^0 continuity with both the surfaces as boundary curve.

In the next sections we analyze in detail the Face-Join operation. In particular we introduce the 1-1 Face-Join operation, the 1-n Face-Join operation and finally the n-m Face-Join.

5.3 1-1 Face-Join Operation

The 1-1 Face-Join operation is the basic tool to define the 1-n Face-Join and the n-m Face-Join.
Joining two surfaces produces a connected surface where the two identified edges are adapted and connected. Often, when the Face-Join is applied, one surface is fixed and one is modified.
Let's define the Face-Join operation in a classic environment between B-Rep faces.

Definition 5.1 (1-1 Face-Join operation)**.** *Given two couples* (A, e_A) *and* (B, e_B) *where* A *and* B *are two surfaces delimited by closed boundaries* W_A *and* W_B*, respectively, and* e_A *and* e_B *are two edges of* W_A *and* W_B*. A* 1-1 Face-Join operation *between* A *and* B *along* e_A *and* e_B *(called shared edge) creates a connected surface* C *where one of the following three cases is verified:*

. *Case I:* A *is fixed and* B *is modified such that* e_B *is attached to* e_A

. *Case II:* B *is fixed and* A *is modified such that* e_A *is attached to* e_B

. *Case III: a new edge* e_C *is defined, both* B *and* A *are modified in order to attach both* e_B *and* e_A *to* e_C*.*

Fig.5.5(a) illustrates an example of 1-1 Face-Join where the first surface, on the right side, is fixed and the other is modified. Otherwise, if both surfaces are modified, a common edge is chosen and then both surfaces are adapted to the chosen edge, thus we can consider the third case as a particular case of the first two cases.
When a 1-1 Face-Join operation is performed it is necessary to specify a

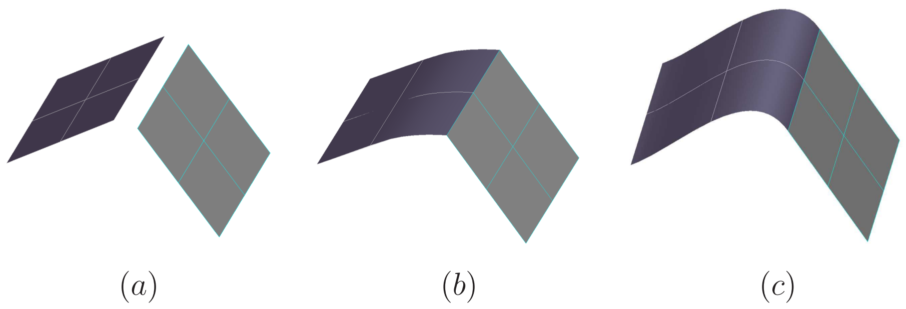


Figure 5.5: Example of join operation: a) surfaces, b) C^0 1-1 Face-Join, c) G^1 1-1 Face-Join. Faces are plotted with $3 \times 3gc$

level of regularity along the shared edge, that is C^0 or G^n continuity, with $n = 1, 2, \ldots$. The regularity of the joining determines the smoothness of the resulting surface. In our work we consider only C^0 and G^1 continuity joins. In case of C^0 continuity, the join gives as result a surface C with points along the shared edge that have two distinct tangent planes. Fig.5.5(b) illustrates an example of two NURBS surfaces joined with C^0 regularity. As we can notice, points on the shared edge have two distinct tangent planes, so the resulting surface is not differentiable everywhere.
Surfaces that are G^1 joined will have tangent vectors with the same direction but with different lengths along the shared edge. Fig.5.5(c) illustrates an example of two NURBS surfaces joined with G^1 regularity. As we can notice, the tangent plane is well-defined for all points of the shared edge.

5.4 1-1 Face-Join for EB-Reps

In order to realize the Face-Join between EB-Reps it is necessary to define the join between a NURBS surface s and a surface described by a Mesh-Face M. The three cases introduced in the previous section are still valid and are formulated in the following form:

- Case I: M is fixed, s is modified,

. Case II: s is fixed, M is modified,

. Case III: both s and M can be modified.

Similarly to the classic case, Face-Join between EB-Rep can be performed as a sharp joint or a smooth one. A sharp joint modifies a surface, the one that is not fixed, so that the faces join with C^0 continuity, adapting the free face to the fixed one.
Instead, a smooth join between EB-Reps requires that the resulting surface is AG^1 or G^1-AE along the shared edge.
In the next subsections we analyze all the cases introduced distinguishing between sharp and smooth join.

5.4.1 1-1 Face-Join C^0

Let's analyze the simplest case: joining a NURBS surface and a Mesh-Face with C^0 continuity. Face-Join C^0 is realized joining two edges respectively on the two surfaces and modifying one or both the surfaces.
To introduce our methods, we assume to have a Mesh-Face $M = (V, E, F)$ and to select a polyline p, defined by m edges of E, delimited by two different vertices (v_0,$v_1 \in V$) on the boundary of M. Moreover we assume to have a NURBS surface s with a NURBS boundary curve c of degree g, defined by b control points and delimited by two control points $\tilde{v}_0$ e $\tilde{v}_1$.
Generally p and c do not have the same number of vertices and control points. Our methods connect the first/last vertices (v_0,$v_1 \in V$) of the Mesh-Face with the first/last points of the NURBS curve $(\tilde{v_0}, \tilde{v_1})$. It is possible to generalize the methods to consider the join along only a piece of p or c.

Case I: Mesh-Face fixed. Given M and s, the NURBS s and its boundary curve c are adapted to join exactly M along p. In particular we apply the "*Adapt NURBS to Mesh*" (ANTM) algorithm that consists in the following steps:

. Multiple knot insertions are applied in order to subdivide c in a piece-wise curve with m linear pieces.

. The control points of c are moved in order to match polyline p. In particular control points that are exactly on c are matched with the correspondent vertices of p, while the control points that are not on c are moved onto the correspondent segment delimited by two consecutive vertices of p. Thus c is completely matched with p.

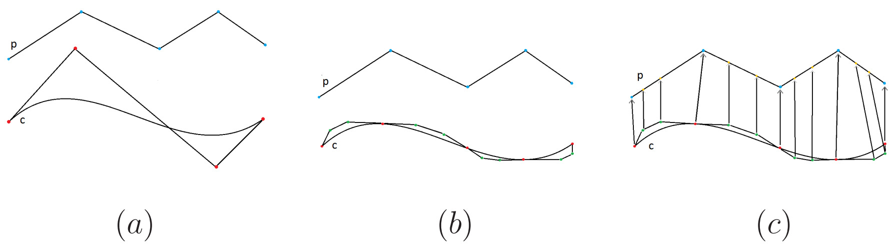


Figure 5.6: ANTM algorithm: a) curve c and polyline p, b) multiple knot insertions applied to c, c) control points of c are attached to points on p.

The steps of the algorithm are illustrated in Fig.5.6. In Fig.5.6(a) the polyline p and the NURBS curve c, with its control points, are illustrated. Fig.5.6(b) shows the result of the first step of ANTM algorithm. Fig.5.6(c) shows how the new control points are matched with the correspondent vertices and points on p. In particular the control points on c (red) are attached to the vertices of p (blue), while the control points that are not on c (green) are attached to points on the segments of p (yellow) according to the second step of the ANTM algorithm. An example of 1-1 Face-Join with the Mesh-Face fixed is illustrated in Fig.5.7 where the NURBS boundary is adapted to the Mesh-Face vertices. As we can notice, the Mesh-Face on the left is unchanged. On the contrary the control point grid associated to s has been modified and the first row has been adapted to the vertices of p.

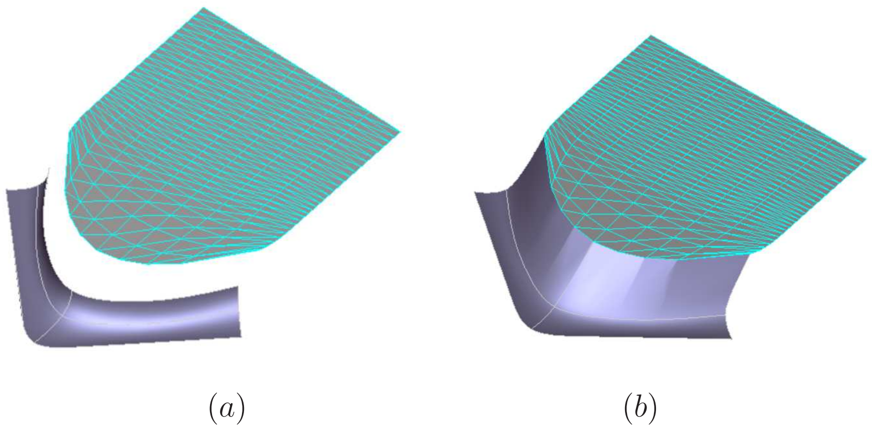


Figure 5.7: Example of Join C^0, Case I: a) Mesh-Face (top) and NURBS surface (bottom), b) result of C^0 join. Faces are plotted with $3 \times 3gc$

Case II: NURBS fixed. Given M and s as above, in order to join the Mesh-Face and the NURBS surface a blending surface s_M is created. In particular our method creates a blending NURBS surface such that:

. one boundary coincides with c,

. the opposite boundary matches the polyline p of M,

. the degree is the degree of c in one direction and 1 in the other direction.

s_M connects all the points $v_1, \ldots, v_n$ on the Mesh-Face to the correspondent $\tilde{v}_1, \ldots, \tilde{v}_n$ control points on the NURBS surface.
Finally a tessellation algorithm is applied to the blending surface in order to create an extension of the mesh M. This tessellation is performed in order to perfectly match p and to match c with AG^0 continuity. An example is illustrated in Fig.5.8. As we can notice, the blending NURBS surface, represented in yellow, is added in order to connect the two surfaces thus forming a watertight C^0 connection. s_M has a structure compatible with both s and M.

Remark 1. The proposed C^0 join methods produce good results in case the Mesh-Face is a low or medium resolution Mesh-Face. However, in case M is a

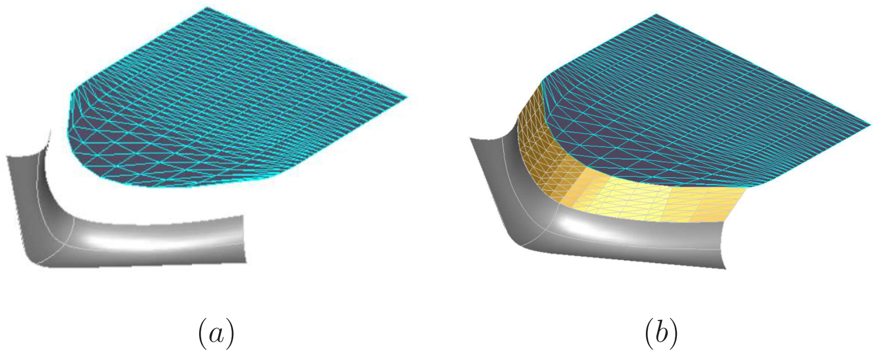


Figure 5.8: Example of 1-1 Face-Join C^0 Case II: a) NURBS surface (bottom) and Mesh-Face (top), b) result of 1-1 Face-Join C^0. Faces are plotted with $3 \times 3gc$

high resolution Mesh-Face the C^0 join methods could be extremely expensive in terms of knot insertions to modify s.
A possible solution is to join s with a low resolution approximation of p.
A 1-1 Face-Join of M with s, considering p_m as the new boundary of M, produces a result that is not C^0 or watertight. The subset of m vertices can be chosen in order to approximate the mesh with AG^0 continuity according to a given tolerance. This kind of joining is really interesting for some engineering applications.

5.4.2 1-1 Face-Join G^1 Almost Everywhere (G^1-AE)

1-1 Face-Join G^1-AE joins a NURBS surface s and a Mesh-Face M with G^1-AE continuity defined in chapter 2. We propose a method to solve Case I where the NURBS s is modified in order to join with G^1-AE continuity the fixed Mesh-Face M.

Case I: Mesh-Face fixed. Given s and M as illustrated in Fig.5.9(a), and a curve c and a polyline p, respectively on s and M, along which the surfaces are joined, the Face-Join G^1-AE method follows these steps:

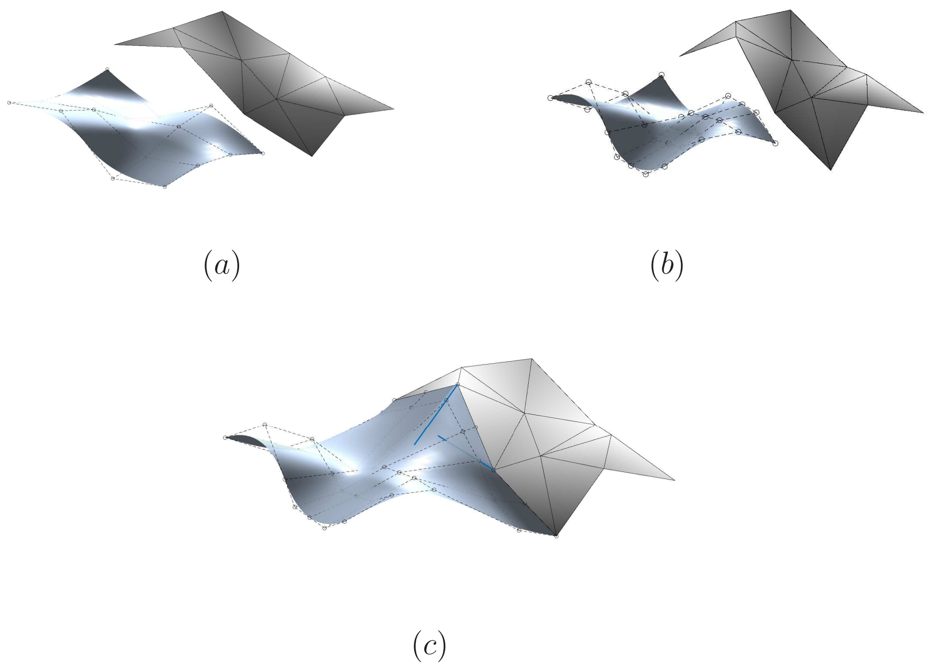


Figure 5.9: Example of join G^1-AE. a) NURBS surface (left) and Mesh-Face (right), b) NURBS structure is adapted to Mesh-Face adding control points, c) Result of join G^1-AE.

. *ANTM* algorithm is applied to the control points of s as shown in Fig.5.9(b),

. internal control points of s are modified according to G^1 connection conditions.

In Fig.5.9(c) the result of join is illustrated. Moreover we notice that NURBS points $p_1, \ldots, p_n$ are only connected with the vertices of the Mesh-Face but they are not G^1 connected because they have two distinct tangent planes associated. Vectors in blue show the distinct tangent planes associated with points $p_1, \ldots, p_n$.

5.4.3 1-1 Face-Join AG^1

1-1 Face-Join AG^1 joins a NURBS surface s and a Mesh-Face M with AG^1 continuity. Also in this case we distinguish between Case I and Case II.

Case I: Mesh-Face fixed. In this case the proposed methods create a strip surface s^* that approximates with AG^1 continuity the Mesh-Face. Then s^* is joined G^1 with s. In particular, given p, a boundary polyline of M, and c, a boundary curve of s, the Face-Join AG^1 methods follow these steps:

. the vertices necessary to create the strip surface starting from the boundary of M are determined.

. a strip surface s^* that approximates the boundary of M is created.

. s^* and s are joined with G^1 continuity using the standard methods for G^1 NURBS join.

s^* is characterized by:

. u degree of s^* is the degree of c,

. v degree is free,

. s^* in u direction is compatible with both boundaries c and p.

In a 1-1 Face-Join it is often necessary to join only a part of the boundary of the Mesh-Face, thus the strip surface is created considering only vertices near to the interested boundaries. If instead the join involves the entire closed boundary of the Mesh-Face, the strip surface is created as a periodic surface. The accuracy of the 1-1 Face-Join AG^1 methods depends on how s^* is created. We propose two different methods to create the s^* structure:

1) A Least Square Strip Surface

2) An AG^1 Approximating Strip Surface

1) Least Square Strip Surface The first and most intuitive method to create s^* approximating the mesh boundary is to select m sets $V_1, \ldots, V_m$ of adjacent vertices of M, with $m > 1$ and construct m least square approximating NURBS curves $c_1, \ldots, c_m$.
The most critical step to be performed is the selection of the m sets of points. In order to explain how to create these sets we need the following definition.

Definition 5.2. *Given a mesh M with boundary W and boundary vertices $v_1, \ldots, v_n$, a vertex v is said to be at* distance d *from the boundary if the shortest path between v and a vertex on W passes throught d edges.*

Vertices on the boundary of the mesh are the first set of points V_1. Then, the second set V_2 is created considering and ordering all vertices with distance 1 from the boundary.
All sets of vertices $V_i,\ i = 3, \ldots, m$ are created similarly as V_2.
When the sets of vertices are created, the least square approximating NURBS curves are computed. Degree g, and knots vector K of the curves are fixed when the first curve c_1 is created considering the set of boundary points. All other curves $c_i,\ i > 1$ are created with the same structure, approximating the associated set of vertices V_i.
The number of control points that determines the number of elements in K is chosen in order to have an AG^1 join between s^* and M. The parametric points for the Least Square method are computed using the following formula:

$$\tau_1 = 0$$
$$\tau_i = \tau_{i-1} + dist_i \qquad \forall i = 1 \ldots n$$

where $dist_i = ||v_i - v_{i-1}||_2$. The parametrization vector has elements

$$t_i = \tau_i / \tau_n \qquad \forall i = 1 \ldots n.$$

We construct the matrix A, whose elements are the B-Spline basis functions of degree g evaluated at the t_i, and the vector $V = (V_x, V_y, V_z)$ of points to be approximated. Then the three systems of normal equations

$$A^T Ax = A^T V$$

are efficiently solved using the Cholesky factorization for any V_x,V_y,V_z vector. Once all the Least Square curves have been created, s^* is created as the sweep NURBS surface that interpolate the created curves. s^* satisfies the following properties:

. degree g in u direction

. knot vector K in u direction

. uniform partition of $[0, 1]$ in v direction as knot vector

. the control points grid is obtained by the control points of curves $c_1, \dots, c_m$ coherently ordered.

An example of strip surface created considering 2 rings around the boundary of a Mesh-Face is illustrated in Fig.5.10. The Mesh-Face with the boundary in red and the strip NURBS surface obtained applying the LS method are illustrated in Fig.5.10(a) and Fig.5.10(b), respectively. Fig.5.10(c) shows how the strip fits the Mesh-Face.

2) AG^1 approximating Strip Surface An alternative method to create a strip surface approximating the mesh boundary is based on the definitions of AG^0 and AG^1 continuity. In this case s^* is created in order to have the same tangent planes of M according to the tolerance of the AG^1 continuity. In particular, the curve on the boundary of M is created using the Least Square approximating method introduced before.
Fig.5.11 illustrates an example of AG^1 approximating strip surface. Fig.5.11(a) and Fig.5.11(b) show respectively the open mesh with the boundary in red and the blending NURBS surface obtained by applying the method. Fig.5.11(c) shows how the blending surface fits the open mesh. Here the blending surface

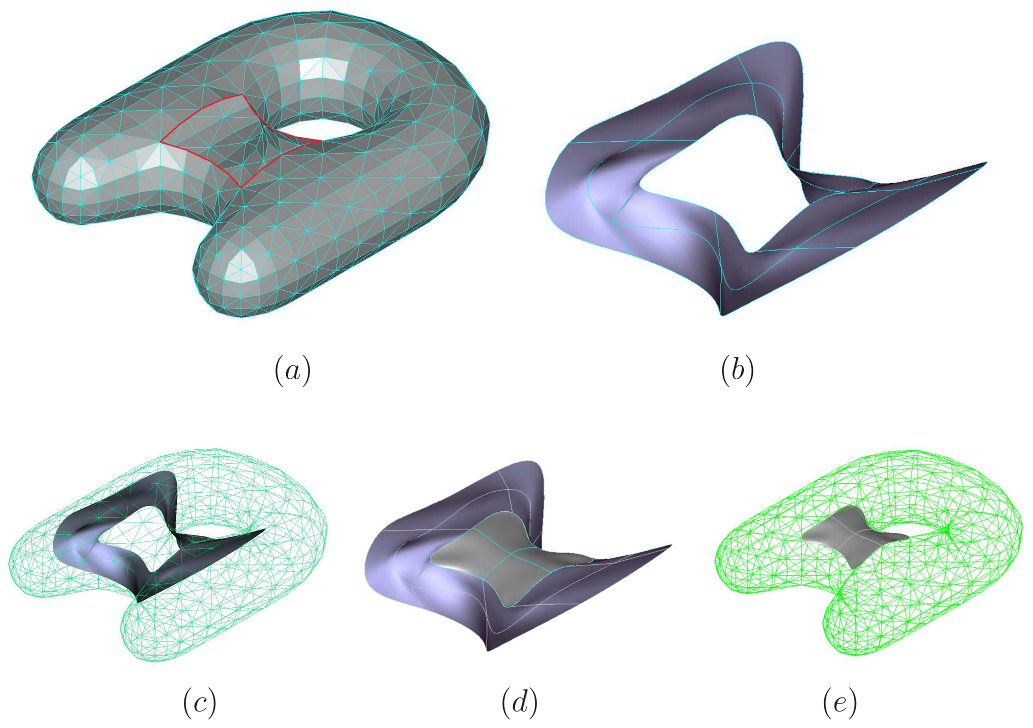


Figure 5.10: Least Square Strip NURBS surface approximating the boundary of the A Mesh-Face. a) Mesh-Face with boundary, b) Least Square Strip, c) strip fitting Mesh-Face, d) NURBS surface joined G^1 with the strip, e) example of AG^1 Face-Join between a NURBS surface and a Mesh-Face. Faces are plotted with $3 \times 3gc$

depends on tolerances chosen for AG^1 and AG^0 continuity. A lower tolerance set of points produces the best blending surface fitting.

Once s^* is created, a G^1 join with s is performed. Two examples are illustrated in Fig.5.10(d-e) and Fig.5.11(d-e) respectively. In both cases the NURBS surface s is joined with AG^1 continuity with M performing a G^1 join with the strip surface s^*. In particular Fig.5.10(d) and Fig.5.11(d) show the result of G^1 join between s and s^*, while Fig.5.10(e) and Fig.5.11(e) show how the modified NURBS surface s fits M.

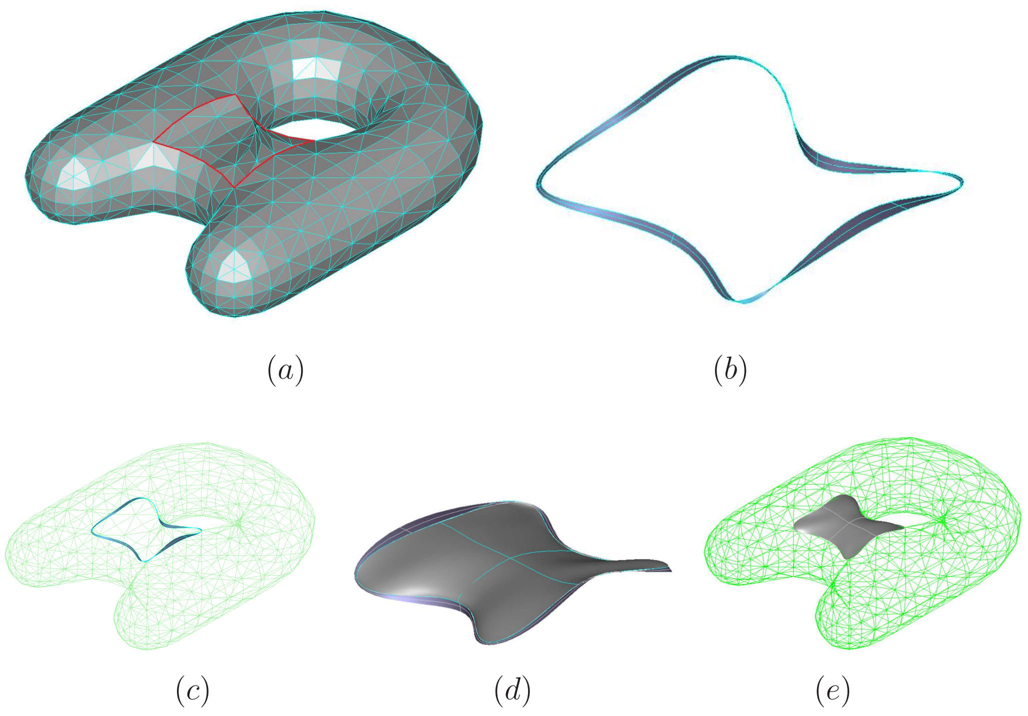


Figure 5.11: AG^1 Strip NURBS surface approximating the hole of the A mesh face a) Mesh-Face with boundary, b) Least Square Strip Surface, c) Strip Surface fitting Mesh-Face, d) NURBS surface joined G^1 with the strip, e) example of AG^1 Face-Join between a NURBS surface and a Mesh-Face. Faces are plotted with $3 \times 3gc$

Case II: NURBS fixed. Given M and s as above, in order to join M and s we create a blending NURBS surface s_M such that:

- one boundary coincides with c,
- the opposite boundary coincides with p,
- it is G^1 joined with s,
- it is AG^1 joined with M.

Finally a tessellation algorithm is applied to the blending surface in order to create an extension of the mesh M. This tessellation is performed in order

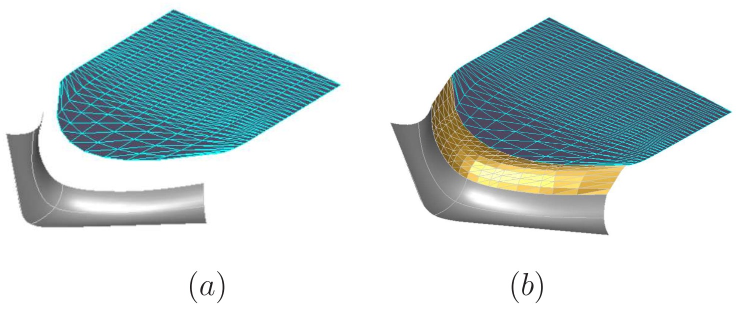


Figure 5.12: Example of 1-1 Face-Join G^1 Case II: a) Mesh-Face and NURBS surface, b) result of 1-1 Face-Join G^1. Faces are plotted with $3 \times 3gc$

to perfectly match p and to match c with AG^1 continuity. The example illustrated in Fig.5.12 shows how the blending surface is created and then tessellated. We notice that the NURBS blending surface joins AG^1 with the Mesh-Face M and G^1 the NURBS s.

5.5 1-n Face-Join between EB-Rep

The 1-n Face-Join operation closes a hole of an open EB-Rep solid with a face. We denote with F the face and suppose it has a close outer boundary W. Instead, the open solid C is supposed to have a bounded hole H which can be shared by more than one face of C. Following the classification given for Face-Join, we have three cases:

Case I: C is fixed, F is modified

Case II: F is fixed, C is modified

Case III: C and F can be both modified

In this section we suppose that the entity to be modified is represented by NURBS surfaces. In particular, in Case I we suppose that F is a NURBS surface and C is an EB-Rep with Mesh-Faces around the hole. In Case II we suppose that F is a Mesh-Face and C is an EB-Rep with NURBS surfaces around the hole.

5.5.1 1-n Face-Join C^0

Our method joins with C^0 continuity a single face F and an open solid C with a hole H. In case F is modified, every edge is adapted to the correspondent edge on H. On the contrary, if C is modified, every surface with an edge or a vertex on H is modified and adapted to the correspondent edge. The method requires the selection of two sets of vertices $VW = (v_{w1}, \ldots, v_{wn})$ and $VH = (v_{h1}, \ldots, v_{hn})$ with the same number of elements. If the selection of vertices does not create the same number of edges, couples of edges are connected or split until the same number of edges on the face boundary W and on the hole boundary H is reached. Connection or splitting process ends when the following conditions are satisfied:

. H and W have the same number of edges

. every edge in H and its correspondent edge in W are delimited by correspondent vertices on VC and VH respectively.

In the next subsections we suppose that H and C respect both conditions. The match between the sets VW and VH completes the 1-n Face-Join C^0 operation.

Case I: Join by modifying F. The method modifies the boundary control points of the NURBS surface F, adapting them to the points of H. In particular, we can subdivide our method in the following steps:

. The structure of F is made compatible with the structure of every edge on H. In particular, for every edge $e_{Hi} \in H$ it is necessary to have the same number of curve segments of the correspondent edge on W, the boundary of F. Knot insertion operations are applied to make this compatibility.

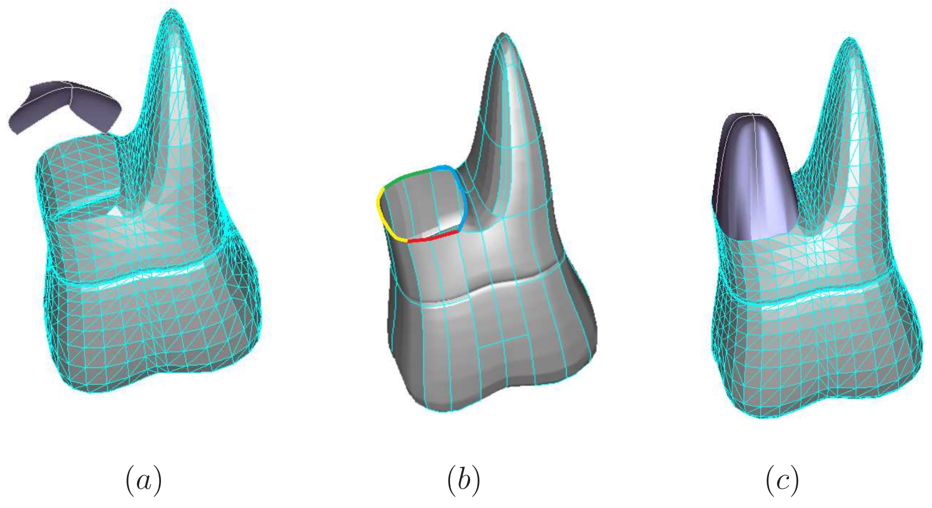


Figure 5.13: Example of 1-n Face-Join Case I. a) NURBS surface above the open EB-Rep, b) Topology of the open EB-Rep c) Result of 1-n Face-Join. Faces of the EB-Rep are plotted with $3 \times 3gc$

. The boundary control points of F are modified to match exactly every edge of H.

This method is an extension of the 1-1 Face-Join C^0 method. In particular the main idea of 1-1 Face-Join C^0 is repeated for every couple of correspondent edges. Fig.5.13 shows an example of 1-n Face-Join C^0. A NURBS surface is joined to an open EB-Rep with Mesh-Faces representing a tooth. The hole on the mesh is closed with a watertight C^0 join. Fig.5.13(a) shows the NURBS surface positioned above the open EB-Rep. At the beginning, the boundary W of the surface does not have the same number of vertices of the hole H. Only 4 vertices on both boundaries are chosen and edges of H are attached in order to obtain the same number of edges of W. Then joining is performed attaching the NURBS curves to the polylines on H. Fig.5.13(c) shows the obtained result.

Case II: Join by modifying C. The method modifies all the faces on H adapting them to the edges of W. More in details, we determine the faces of the EB-Rep C which share either edges on H, or only a simple vertex on H. Then for every face, the associated grid of control points is modified. The method can be described with the following steps:

. For every face with an edge on H to be modified, the 1-1 Face-Join C^0 is applied.

. For every face with a single vertex to be modified, the interested control point in the NURBS grid and the edges delimited by the vertex are changed.

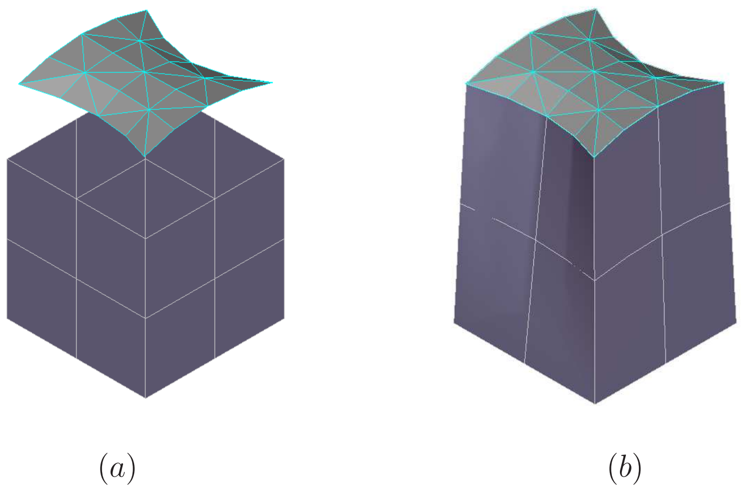


Figure 5.14: Example of 1-n Face-Join, Case II: a) Mesh-Face above the open EB-Rep, b) Result of 1-n Face-Join. Faces are plotted with $3 \times 3gc$

An example is illustrated in Fig.5.14. A Mesh-Face is joined with an open EB-Rep model described by NURBS faces. Fig.5.14(a) shows the Mesh-Face placed above the open solid.
Fig.5.14(b) shows the obtained result. We notice that all surfaces around the hole are adapted to fit the surface. In this case the boundary W of the surface does not have the same number of vertices of the hole H. In particular W has 4 edges, while H has 16 edges. Only 4 vertices on both boundaries are chosen and the edges of H are attached in order to obtain the same number of edges of W.

5.5.2 1-n Face-Join AG^1

The 1-n Face-Join AG^1 closes the hole of an open solid with a surface creating a smooth join. The methods have the same structure of the 1-1 Face-Join AG^1. We distinguish between Case I and Case II.

Case I: Join by modifying F In this case F is a NURBS surface, while C is an open solid with a hole H and n Mesh-Faces around H.
A set of strip surfaces $\{s_i^*\}_i^n$, approximating the Mesh-Faces around H is created. Then a G^1 join between $\{s_i^*\}_i^n$ and F is performed.
The method can be structured in the following steps:

. For every Mesh-Face around H an approximating strip NURBS surface s_i^* is created.

. Adjacent strip surfaces s_i^* and s_j^* are joined in order to have G^1 continuity.

. The set $\{s_i^*\}_i^n$ is joined G^1 with F following the existent methods for NURBS joining.

Creation of every $s_i^* \quad i = 1, \dots, n$ has been described in the previous sections for the 1-1 Face-Join AG^1.
Once all $s_i^* \quad i = 1, \dots, n$ are created, join between NURBS entities is performed following algorithms described in [24].
In this case it is necessary to consider also the algorithms for G^1 connection of three and four NURBS surfaces ([24]).

Case II: Join by modifying C In this case F is a Mesh-Face, while C is an open solid with a hole H and n NURBS faces around H.
A strip surface s^* approximating the Mesh-Face is created. Then a G^1 join between s^* and the NURBS faces around H is performed.
The method can be structured in the following steps:

. A strip NURBS surface s^* approximating F is created.

. s^* is joined G^1 with all the NURBS faces around H following the existent methods for NURBS joining.

The procedure for creating every s^* has been described in the previous sections for the 1-1 Face-Join AG^1.
Once s^* is created, join between NURBS entities is performed following algorithms described in [24].
Also in this case it is necessary to consider also the algorithms for G^1 connection of three and four NURBS surfaces ([24]).

5.6 n-m Face-Join between EB-Rep solids

The n-m Face-Join between EB-Rep solids is the most general case of join. In order to describe n-m Face-Join between EB-Reps, we introduce the following definition.

Definition 5.3 (n-m Face-Join between solids)**.** *Given two solids A and B with two holes H_A and H_B delimited by two closed boundary W_A and W_B respectively on A and B. A n-m* Face-Join between solids *between A and B along W_A and W_B creates a connected solid C such that A and B are joined closing the holes with C^0 or G^1 continuity.*

Fig.5.15 illustrates an example of different results of Join operation. A teapot body and its spout, illustrated in Fig.5.15(a), are joined in two different ways. Fig.5.15(b) shows the result of a sharp join, while Fig.5.15(c) shows the result of a smooth join.
Join differs from Union boolean operation because both solid shapes can be modified during the operation to adapt each other in the desired way. The boundary of the modified solid is connected with the boundary of the fixed solid, closing both the holes.
In this section we analyze the n-m Face-Join between two Extended B-Reps $E1$ and $E2$. We propose to modify the entity described by NURBS surfaces, leaving unchanged the one described by Mesh-Faces. We suppose that the

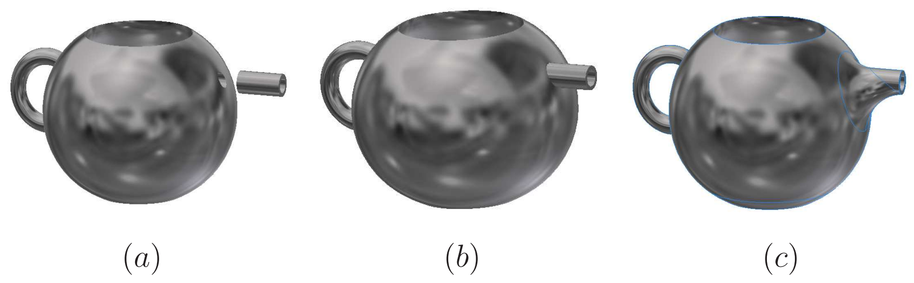


Figure 5.15: Example of join operation: a) body of the teapot and spout, b) union between objects, c) join between objects

open solid $E1$ has a hole delimited by a wire $W1$, while $E2$ is delimited by $W2$ and $W1$ and $W2$ do not necessarily have the same number of edges. Obviously an EB-Rep can have different kinds of faces around a hole. We focus only on the one involving Mesh-Faces and NURBS because all the most used analytic surfaces can be represented exactly by a NURBS surface. A n-m Face-Join between a NURBS and an analytic surface is reduced to a join between two NURBS, which is a case solved in classic B-Rep systems. Moreover n-m Face-Join between a Mesh-Face and an analytic surface is reduced to a join between a NURBS and a Mesh-Face.

5.6.1 n-m Face-Join C^0

The n-m Face-Join method attaches with C^0 continuity $E1$ and $E2$ and is based on the 1-n Face-Join C^0 algorithm. More in detail, the method joins couples of correspondent surfaces closing both the holes of $E1$ and $E2$. It can be described with the following steps:

. Split/Join edges of $W2$ in order to have the same number of edges of the wire $W1$.

. For every face with a vertex on $W1$, the 1-1 Face-Join C^0 method is applied.

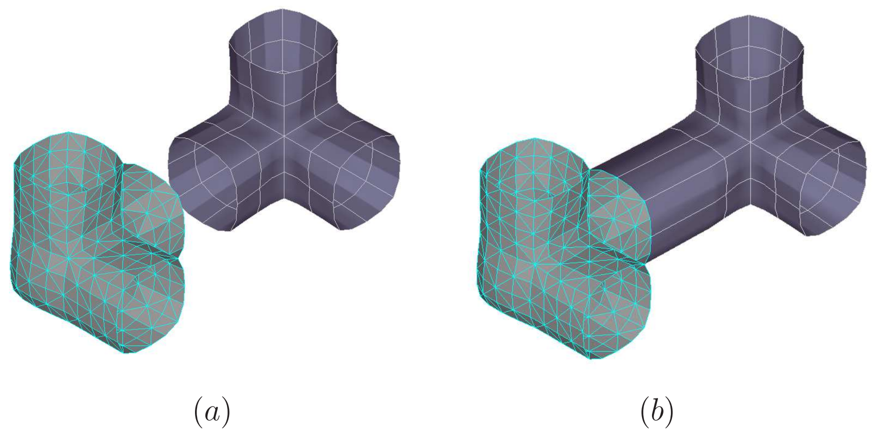


Figure 5.16: Example of n-m Face-Join C^0: a) an EB-Rep "Tube" with Mesh-Faces and an EB-Rep "Tube" with NURBS faces. b) Result of n-m Face-Join C^0. Faces are plotted with $3 \times 3gc$

Fig.5.16 shows an example of C^0 join between open solids described by EB-Reps made of Mesh-Faces and NURBS surfaces respectively. In Fig.5.16(a) the two solids are positioned such that there is an empty space between them. Fig.5.16(b) shows the result of EB-Rep join C^0. We can notice how the boundary faces of the NURBS tube are extended in order to close the nearest hole of the mesh tube. Correspondent vertices selected at the beginning are attached and edges are modified in order to have a watertight joining.

5.6.2 n-m Face-Join AG^1 of Open Solids

The n-m Face-Join AG^1 method attaches with AG^1 continuity two solids $E1$ and $E2$ and is based on the 1-n Face-Join AG^1 algorithms. Also in this case it is necessary to consider also the algorithms for G^1 connection of three and four NURBS surfaces. In general G^1 connection of $n \geq 3$ NURBS surfaces sharing a common vertex has to be considered ([24]).

Chapter 6

Finite Cell Method applied to Extended B-Rep solids

Computer-aided engineering (CAE) is the usage of computer software to aid in engineering analysis tasks. CAE tools are used, for example, to analyze the robustness and performance of components and assemblies. The term encompasses simulation, validation, and optimization of products and manufacturing tools.
The most famous computational tool for performing engineering analysis is the Finite Element Analysis (FEA), that divides a complex problem into small elements. In applying FEA, the complex problem is usually a physical system with a mathematical equation associated, while the divided small elements of the complex problem represent different volumes in the physical system.
Various methods has been introduced and improved. Actually problems arise, for example, when considering heterogeneous materials or more generally when discretizing structures which have a very complex geometry which might even change during the computation. To overcome these problems the Finite Cell Method (FCM) [26] was introduced. It can be considered as a combination of a fictitious domain method with high-order finite elements.
In this chapter we illustrate in detail an example of FCM applied to an EB-

Rep representing a perforated tooth before being incapsulated. Using Finite Cell Method we simulate the application of a force on a specific point of the tooth and analyze the effects of this force on the object.
In the first part of the chapter we introduce the Finite Cell Method describing its features. Then we explain how the solid virtual object has been created. In the last part we illustrate results and discuss the use of extended solids for physical analysis.

6.1 Finite Cell Method

Finite element analysis (FEA) with standard methods requires the discretization of the physical domain into a finite element mesh, whose boundaries respect the boundaries of the starting geometry. This constraint is still a severe bottleneck in the simulation pipeline, in particular, when highly complex geometries need to be dealt with. All the new methods introduced in the last decade have the main goal to alleviate or eliminate the discretization challenge for complex geometries. The most well-known concept is most probably Isogeometric Analysis, introduced by Hughes and co-workers in 2005 [37] in the context of CAD. The main idea of Isogeometric Analysis is to use the same higher-order smooth spline basis functions for the representation of geometry in CAD and the approximation of solutions fields in finite element analysis.
In cases where no CAD model is created beforehand, Isogeometric Analysis cannot be applied, and a more general approach is required. FCM is introduced by Parvizian, Duster and Rank in 2007 ([26] and [27]) as a solution for these cases. The main idea of this type of method consists of the extension of the physical domain of interest beyond its potentially complex boundaries into a larger embedding domain of simple geometry. An example is given in Fig.6.1 where a *potato* domain is extended with a fictitious domain creating a simple area that can be approximated with a regular mesh. The fictitious area is penalized by a coefficient α that is near to 0. Doing so, a simple

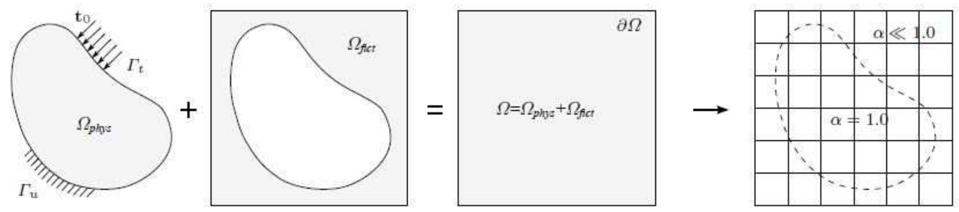


Figure 6.1: Fictious domain theory

structured grid is used for the discretization. This grid eliminates the time consuming and errors during the generation of boundary conforming meshes. The fictitious domain concept is then combined with higher-order basis functions for the approximation of solution fields, the representation of the geometry at adaptive quadrature points, and weak imposition of unfitted essential boundary conditions. In particular an adaptive quadrature is used in order to increase accuracy of numerical integration obtained by Gauss quadrature. Fig.6.2 illustrates the generation of the sub-cell structure in two dimensions following the general procedure of recursive bisection used for quadtree. Starting from the original finite cell of level $k = 0$, each sub-cell of level $k = i$ is first checked whether it is cut by a geometric boundary. If true, it is replaced by four equally spaced cells of level $k = i + 1$, each of which is equipped with $(p + 1) * (p + 1)$ Gauss points. Partitioning is repeated for all cells of current level k, until a predefined maximum depth $k = m$ is reached. In 3D the sequence is analogous, with the difference that every cell is a cube and is subdivided in 8 octants, as in the octrees structure.

The corresponding algorithms are simple, accommodate geometries of arbitrary complexity, and allow for reliable automation of the discretization process. The finite cell method maintains optimal rates of convergence with mesh refinement and exponential rates of convergence with increasing polynomial degree, and thus guarantees full accuracy at a moderate computational effort [46]. Due to the flexibility of the quad-based geometry approximation, the finite cell method can operate with almost any geometric model, ranging

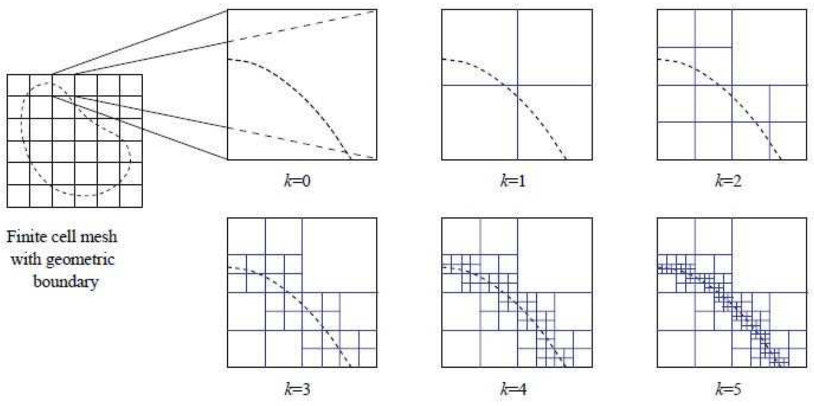


Figure 6.2: Adaptive quadrature of Finite Cell Method

from boundary representations in CAGD to voxel representations obtained from medical imaging technologies.

6.2 FEA and EB-Rep for a Dental Implant application

A dental implant, also known as an endosseous implant, is a surgical component that interfaces with the bone of the jaw or skull to support a dental prosthesis such as a crown, bridge, denture, facial prosthesis or to act as an orthodontic anchor. Fig.6.3 shows an example of dental implant.

The basis for modern dental implants is a biologic process called osseointegration where materials, such as titanium, form an intimate bond to bone. As illustrated in Fig.6.4, firstly existing tooth is extracted then, after three months, implant fixture is placed. After about 3 or 4 months the healing abutment is placed and finally, after 2 or 3 weeks the final dental prosthetic is added.

Success or failure of implants depends on a finite number of factors regarding healthy conditions of the person receiving it. It is fundamental to determine exactly the position and the number of implants. In order to do it, it is also

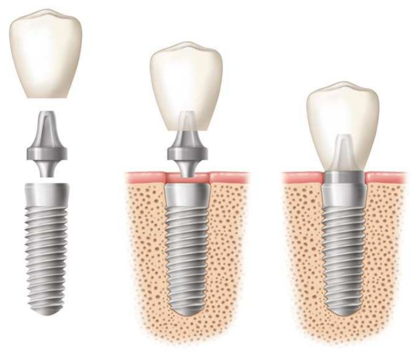


Figure 6.3: Dental implant

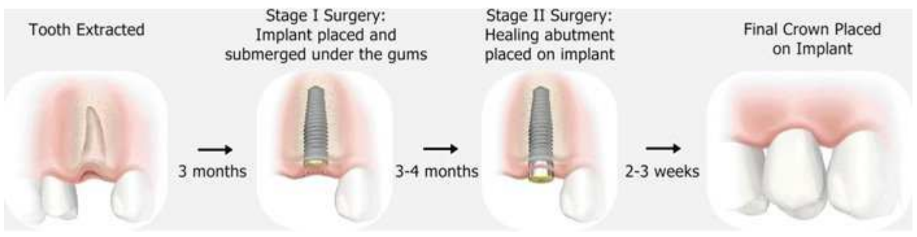

Tooth Extracted
3 months
Stage I Surgery: Implant placed and submerged under the gums
3-4 months
Stage II Surgery: Healing abutment placed on implant
2-3 weeks
Final Crown Placed on Implant

Figure 6.4: Detailed dental implant process

necessary to evaluate the amount of stress that will be put on the implant and fixture during normal function.
Planning the position and number of implants is key to the long-term health of the prosthetic since biomechanical forces created during chewing can be significant. The position of implants is determined by the position and angle of adjacent teeth, lab simulations or by using computed tomography with CAD/CAM simulations and surgical guides called stents.

In our work we want to simulate the stress on a dental prosthetic used for a dental implant.
In particular, we consider the tooth represented by the mesh illustrated in Fig.6.5(a). Then we cut the inferior part of the tooth and created the cavity

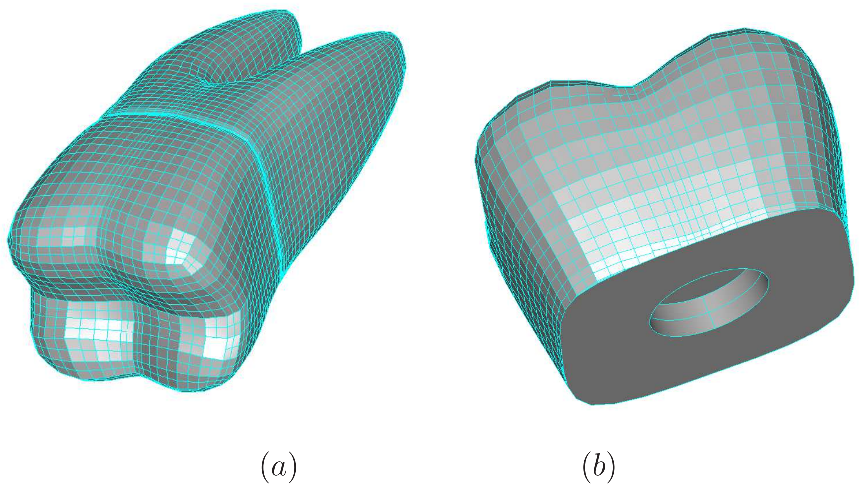


Figure 6.5: a) Original Tooth, b) modeled prosthesis

for the abutment. The resulting model is shown in Fig.6.5(b). In this case we observe that the analysis model is an EB-Rep model, which consists of mesh for the upper part, analytical surface for the cavity (cylinder) and a NURBS surface for the bottom.

Once the model is created, the stress analysis is applied. We simulate the

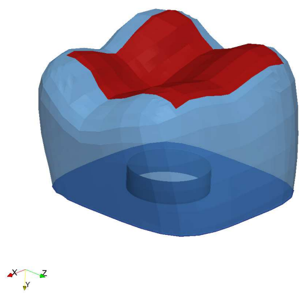


Figure 6.6: Force applied on the tooth

application of a constant force F on the top of the prosthesis. Fig.6.6 shows the force applied on the object. Fig.6.7 shows how the adaptive quadrature is performed on the object. In this case more than 5 refinements are applied to the grid and a discretization is obtained. Then integration is performed and

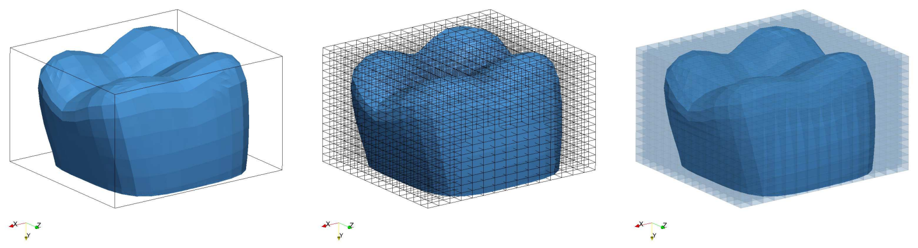


Figure 6.7: Adaptive quadrature refinements

results are shown in Fig.6.8. We can notice how the tooth is deformed and where there is the major stress (coloured in red). In conclusion, we applied a

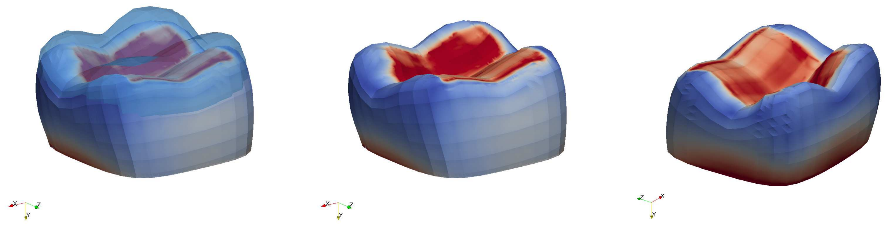


Figure 6.8: Tooth Boundary Conditions and applied Force

stress analysis test to an Extended B-Rep model obtaining satisfying results. Indeed conditions required to perform Finite Cell Method are the possibility to determine the closure of an object, separating the inside from the outside. In this way it is possible to use the fictitious domain. An Extended B-Rep model has a valid representation that separates the inside from the outside of a solid object.
More in general, other analysis methods simply require properties of classic solid object representations. Extended B-Rep models guarantee all these

properties. Future developments will investigate the use of EB-Reps in other interesting FEA examples.

Chapter 7

Research activity performed for Hyperlean

During my P.h.D I collaborated with Hyperlean S.r.l, a spin-off from the Polytechnic University of Marche. Hyperlean [4] is interested in the area of knowledge management software systems to support the product lifecycle. The main software created and improved is LeanCost, an optimal software solution for product costing and cost simulation.
Our collaboration with Hyperlean consisted in the creation of geometric tools currently implemented in LeanCost. The algorithms we created involved the Injection Molding and the Boolean Operations between meshes. In the first case the algorithm has been studied and implemented considering innovative aspects of EB-Reps models. In particular all the steps of the algorithm work on both analytic surfaces and Mesh-Faces. In the second case the algorithm concerning Boolean Operations between meshes integrates the efficient representation of meshes described in chapters 3 and 4. It introduces a new approach for the representation of results of Boolean Operations between meshes that have been efficiently represented in an Solid Modeling System extended to manage both Mesh-Faces and analytic surfaces.
In the first part of this chapter we are going to introduce LeanCost and its peculiarities. Then we explain in detail tools realized to improve LeanCost.

7.1 LeanCost

LeanCost is the optimal software solution for product costing and cost simulation developed by Hyperlean. This software has the main goal to reduce the time spent estimating cost of production of an object and generating offers.
The main idea of the software is to provide different costs of product of an object depending on processes and materials used to realize it. Software is subdivided in two different parts: a section in which costs of processes and raw materials are combined in order to return an economic production cost of the object and a previous geometric section in which feature recognition is applied in order to recognize steps necessary to produce a part starting from a raw material. This second part is the most interesting for our work and the one we are going to consider in detail.
Geometric section considers a 3D CAD model with some given characteristics of the project. A detailed analysis of geometry is performed, extracting features and main operations necessary to realize the model. Then technology to use is determined considering features extracted.

Injection Molding

A section of LeanCost is dedicated to Injection Molding Technology, a method of processing predominantly used for thermoplastic polymers. That technology consists of heating thermoplastic material until it melts, then this melted plastic is forced into a steel mold, where it cools and solidifies.
For our studies what is really important is the Injection Molding process cycle. This cycle consists of three major stages:

- Injection Stage
- Cooling or Freezing Stage
- Ejection and Resetting Stage

In the first stage, the mold is closed and the injection unit facilitates the flow of molten material from the heating cylinder through the nozzle into the

mold. In the second stage molten material is cooled and the object solidifies. In the last stage, the mold is opened, the part is ejected and the mold is then closed again in readiness for the next cycle to begin.
According to this process, we can easily understand that the main cost of Injection Molding is the creation of the mold. The more complicate is the mold, the more expensive is the production of the object. The most common types of molds used in industry are two-piece molds and multi-piece molds. The first category has only one primary parting surface and consists of two major pieces, core and cavity, separated along a single direction to eject the molded part. The second category is used instead to create complex shaped parts that can not be made by two-piece molds. In this case molds have many parting directions.
The Parting Direction is considered as the main direction of ejection for the mold. In our case it is the direction along which it is possible to see the biggest area of the surface of the object.
If we want to estabilish approximatively the complexity of realization of a 3D model using Injection Molding we need to determine some fundamental characteristics of the object and of its mold. In particular it is necessary to:

- determine if the object is moldable or not,
- estabilish which is the principal parting direction,
- find the best solution between all possible molding solutions.

The tool we realized to improve LeanCost computes these three necessary information and allows users of LeanCost to extimate an approximation of production cost of a given 3D model. Our algorithm has been created assembling algorithms that can be applied on EB-Rep models, considering the peculiarities of both analytic surfaces and Mesh-Faces. In the next section we explain in detail the realized tool.

7.2 Compute Best Molding Solution

Our aim is to describe in detail the method we created in order to obtain those fundamental characteristics introduced in the previous section. Our algorithm considers a solid model S and its triangulation M_S and determines if the model is moldable or not. If the solid object is considered as moldable, the principal parting direction is computed and all possible solutions, given a maximum number of pieces n, are computed.
Our algorithm can be subdivided in four principal steps:

- Compute all the candidate extraction directions $d_1, \ldots, d_m$.
- Compute Face Visibility and determine if the object S is moldable.
- Compute the Parting Direction d.
- Compute all solutions $s_1, \ldots, s_t$ which have at maximum n pieces.

We are going to describe all steps in detail in the next paragraphs.

Compute Candidate Extraction Direction

We analyze the solid model S, considering in particular all its faces according to its Boundary Representation. For every face we consider the type of the face, distinguishing between planar and cylindric faces. For every planar face we compute area and the normal vector. Instead for every cylindric face we consider the axis of the cylinder. Then we assemble faces in groups according to their main direction and compute the total area of every group of faces. Mesh-Faces are analyzed considering normals of planar faces. This procedure allows us to estabilish which are directions that are normal or axis of the biggest areas in the solid object. These directions are the main candidate to be the Parting Direction of the mold.
We have these two remarks:

1. Structure of a mold implies that one piece is extracted along Parting Direction d while the second one is extracted along $-d$. So, when a direction d is considered, we automatically consider also $-d$.

2. Solid models are created using CAD systems in which the object is drawn with a corner in the origin and with main faces parallel to the main planes. Considering the main directions x, y, z is often a good choice to find the Parting Direction.

Once the main candidate directions are computed, we order them according to the associated area. The direction that is normal for the biggest area is analyzed as first candidate Parting Direction.

Compute Face Visibility and determine if the object S is moldable

Once possible extraction directions are computed, the face visibility is analyzed for every direction d and its opposite $-d$, in order to understand if the object is moldable or not.
For every face of the solid is determined:

. if the face is a cylinder with axis parallel with d. In this case the face is considered as normal to directions d and $-d$.

. if the face is a planar face with normal vector d and $-d$. In this case the face is considered as normal.

Then, for every face f_i an algorithm is performed that determines visibility of the mesh M_{f_i} associated with f_i. For Mesh-Faces we consider the mesh itself. This algorithm creates a dictionary in which with every face f_i is associated a list of values indicating the visibility of the correspondent triangle in M_{f_i}. Visibility is computed using depth buffer.
Once a dictionary is created for both directions d and $-d$, for every face is determined if it is totally visible, partially visible, or completely not visible from d or $-d$. With this last step faces are grouped in these three categories. Repeating this process for every couple of directions allows us to have precise information on visibility of faces along candidate extraction direction.
In order to estabilish if a part is moldable or not it is enough to control if all the faces of S are completely visible for at least one direction or its opposite. If there are one or more faces that are not visible from any direction the

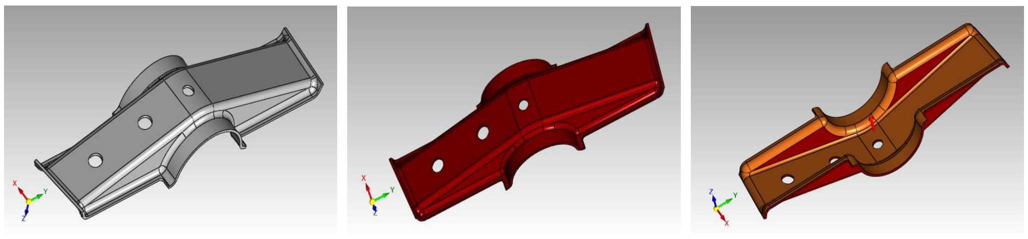


Figure 7.1: Example of Moldable Part and its Parting Direction

object is considered non moldable. On the contrary, if every face is visibile from at least one direction, the object is considered moldable.
An example is illustrated in Fig.7.1(a) where a solid object is represented. This particular part is moldable and we can easily see that one needs a two-piece mold to create the object. Fig.7.1(b) and Fig.7.1(c) show visible faces from the parting direction and from its opposite. Faces coloured in red are faces visible from the Parting Direction, faces coloured in orange are faces visible from opposite Parting Direction.

Compute the Parting Direction

After having determined the moldability of a part, Parting Direction is the direction, and its opposite, that has the biggest visible area. In order to compute this couple of directions, we consider, for every candidate direction d, the total area of faces visible from d and $-d$ and choose the most visible area. Fig.7.2 shows an example of determination of the Parting Direction. Fig.7.2(a) shows the object to be molded, Fig.7.2(b-c-d) show different candidate Parting Directions. From this simple example it is easy to understand that the Parting Direction is the one shown in Fig.7.2(c) because it has the biggest visible area, coloured in red.
Other coloured faces are respectively:

. Not visible faces in cyan

. Faces partially visible from d in yellow

. Faces partially visible from $-d$ in violet

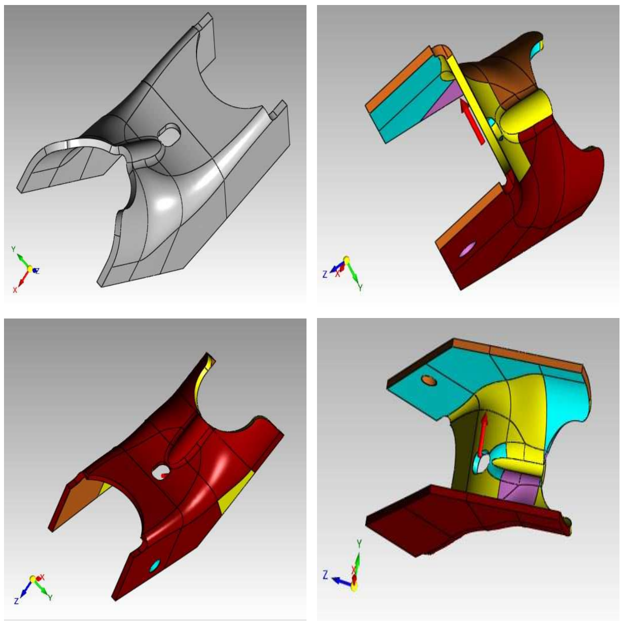


Figure 7.2: Example of candidate Parting Directions: a) object, b-c-d) examples of different candidate parting directions

Compute all solutions which have at maximum n pieces

Once a Parting Direction d_1 is computed, all possible solutions, given a maximum number n of pieces, are computed considering all possibilities. Starting from d_1 and $-d_1$, the second parting direction d_2 is chosen as the one that minimize the remaining non visible area. All directions $d_3, \ldots, d_n$ are computed following the minimization rule. In the last step all solutions are ordered considering the minimum number of pieces necessary to mold the part.

Fig.7.3 shows an example of complete solution. Fig.7.3(a) shows the part to mold. Fig.7.3(b-c) show the Parting Direction. Fig.7.3(d-e-f) show all the other directions necessary to mold all faces of the object. As we can notice existing holes require single pieces to be molded. These additive pieces increase the complexity of the mold.

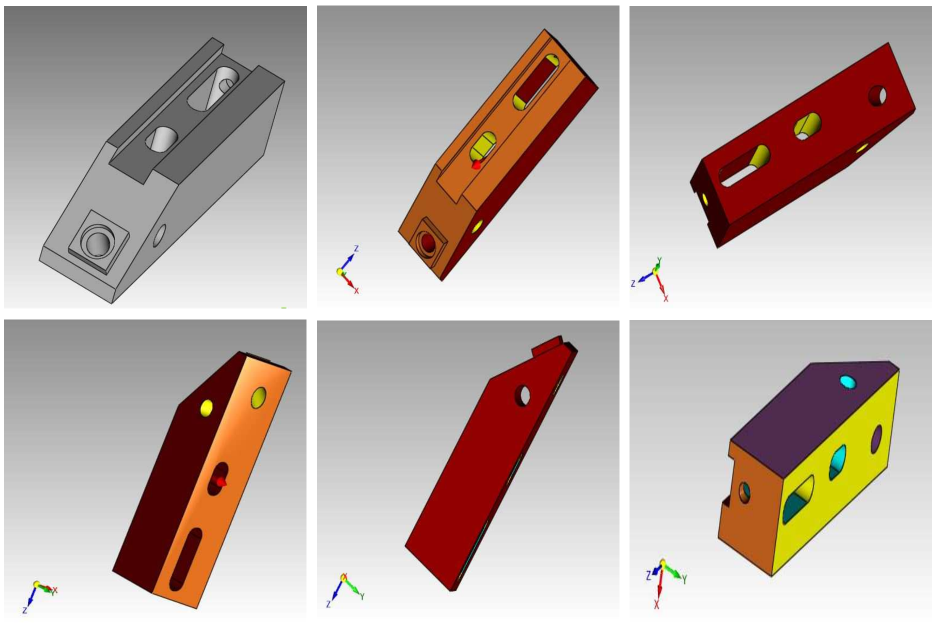


Figure 7.3: Example of Moldable Part Solution. a) object, b-c-d-e-f) parting directions in the solution

7.3 Boolean Operations in LeanCost

Boolean Operations are the basic instruments for Solid Modeling. LeanCost needs to extimate tools necessary to obtain a solid model starting from a raw material. These tools include boolean operations.
Inside LeanCost, for every solid model, a mesh is created in order to visualize the object. Using existing libraries it is possible to perform boolean operations between meshes associated with solid objects.
LeanCost takes in input a solid object and operates on solid objects, while result of a boolean operation applied to meshes is obviously a mesh. It was necessary to give, as result, a B-Rep representation of the resulting solid object. Due to this necessity we devised a method that recreate the wire of the solid object result of the boolean operation between meshes associated with two solid objects.
In the next paragraphs we are going to describe in detail characteristics of the

boolean operations with Carve library and the algorithm created to obtain a solid object as result.

Boolean operations between meshes

The algorithm that we created gives the possibility to rebuild the boundaries associated with the faces of a B-Rep model obtained as the result of a boolean operation between two solids A and B. More in detail, given two solids A and B, MA and MB are defined the meshes associated respectively with A and B and $\bullet$ is called a boolean operation between meshes. MR is the resulting mesh.
The goal of the algorithm is to get from MR the solid R represented by B-Rep. In order to reach efficiently the goal it is necessary to build the mesh MR associated with a solid S building a single mesh for each single face and filletting vertices on the boundary of adjacent faces. In this way, each face of the mesh MS has a referring f_i face on S.
It is possible to subvide the algorithm in the following steps:

. Boolean operation is performed and information is extracted.

. Creation of new boundaries of the interested face

. Creation of a B-Rep result

These steps are detailed below.

Boolean operation is performed and information is extracted The first part of this algorithm consists of the execution of the boolean operation $\bullet$ between meshes MA and MB associated with the solids A and B. Meshes have to be structured as follows:

. vector v of the vertices made by $3 * nv$ elements. These elements of v respect the following rule:

  - $v[3i] = mesh \rightarrow v[i] \rightarrow x$
  - $v[3i+1] = mesh \rightarrow v[i] \rightarrow y$

  - $v[3i+2] = mesh \rightarrow v[i] \rightarrow z$

. vector f of the faces of the mesh. Considering only triangular meshes, the structure of f is similar to the structure of v but in this case the indices of the three vertices that represent each face are considered. The rules are:

  - $f[3i] = mesh \rightarrow f[i] \rightarrow v[0]$
  - $f[3i+1] = mesh \rightarrow f[i] \rightarrow v[1]$
  - $f[3i+2] = mesh \rightarrow f[i] \rightarrow v[3]$

. list of attributes to be associate to each face of the mesh. In this case, an internal values list is created in order to give the index related to the face f_j associated with the face in the B-Rep model for each face of the mesh. Infact, as said before, the meshes are built discretizing the B-Rep model associated with the solid. So, each face of the mesh belongs to one and only one face of the B-Rep model.

Once the structure related to both meshes MA and MB is created, the boolean operation is performed and we go on creating the structure needed for the following steps.
First of all, the result mesh starting from the vectors given as output of the algorithm realizing the boolean operation. Then, it is performed a method that gives the possibility to get every intersection polylines from that boolean operation. Being more precise, the algorithm that realize this boolean operation determines all the shared edges of the two involved meshes. These edges are called intersection edges. For each of these edges, the indices of the faces that intersect and the starting and ending points of the edge are kwown.
All the closed intersection polylines are build using this information. Finally, the faces of the B-Rep model directly involved in boolean operation are determined using the information converted in the attributes list.
These face are all and only faces for which the boundary is modified. In addiction, it is determined if a point belongs to an edge of the B-Rep model

face or if it is a point internal to it for each polyline. Then, a list is created in which it is listed for each non internal point its referring edge in the B-Rep model of the considered solid.

Creation of new boundaries of interested faces The main part of the algorithm consists in the creation of new boundaries associated with the faces involved in the boolean operation. This method is iterated on every face belonging to the solids A and B. The algorithm considers all the loops that delimit a face f, independently from the fact that they are internal or external. All the edges of the involved loop are determined. These edges are fundamental for the construction of the new loop. All the polylines p_i that lie completely or partially on the face f are considered. The subpolylines s_j lying on f are determined for each one of these. These s_j are polylines with all the vertices lying on f. s_j can be divided in two types:

. open polylines starting from an edge e_k and ending in an edge e_l where e_k an e_l are not necessarily different

. closed polylines in which all the points are internal to f.

In the first case, the polyline s_j will be a new edge of one of the loop of the face. In the second case, s_j will be a new loop of f. Once all the s_j are determined, the main part of the algorithm is performed: building of new loops. The method is the following:

. the first subpolyline s_1 is considered and the starting and ending edges e_k and e_l are determined, if they exist. More in detail, the point p_{first} and p_{last} in which s_1 intersects respectively e_k and e_l are determined. If e_k and e_l do not exist and s_1 is internal, a new loop is added to the face. If e_k and e_l exist, it is necessary to determine the following edges making the loop. Considering p_{last} an the side e_l related to it, it is possible to distinguish three different cases:

  - the new edge to be built is delimited by another subpolyline s_j with starting or ending point on that side.

- the new side to be built is delimited by a vertex native to the side
- if e_k=e_l, p_first and p_last make the ends of the last side necessary to close the loop

Each of these possibilities needs to make checks in order to determine right sequences of the sides delimitating the edge of the face.

. During every check the following point of the loop is determined. The algorithm goes on until the starting point is reached.

First, a rule for the orientation of edges, loops and polylines should be set, in order to semplify a lot the process and in order to make possible to determine always the direction of travel to follow. If this were not done, it should be necessary to make expensive direct controls in order to choose the vertex. It is necessary to make the following considerations to get a correct result:

. It is necessary to set correctly the last point of the external loop. In particular, the last point of the loop is the one from which it is started with the first considered polyline. Fig.7.4 shows an example of a new boundary created with this algorithm. In this case, and also in all the following examples, vertices are considered in counterclockwise order. As we can see, the blue polyline divides the face in two parts, the green polygon delimitates the new face. The first and last point of the boundary is p_1, that is the starting point of the polyline.

. It is possible that the same edge is intersected by two or more different polylines. In this case, it is necessary to determine properly the portion of the edge belonging to the external loop. Fig.7.5 shows an example of an edge intersected by two polylines. In this case, following the counterclockwise order, we obtain the new boundary delimited by the green line. If there was no order, we should have established which was the order of points considering directly triangles of the result mesh.

. It is possible that a polyline starts from one vertex on the external loop and ends on one edge belonging to an internal loop. In this case, the

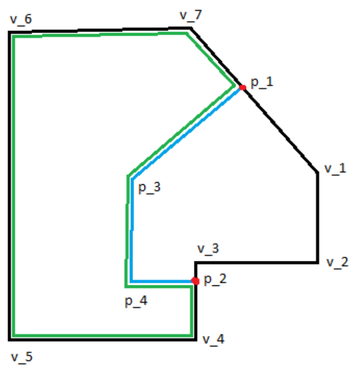


Figure 7.4: Example of new boundary

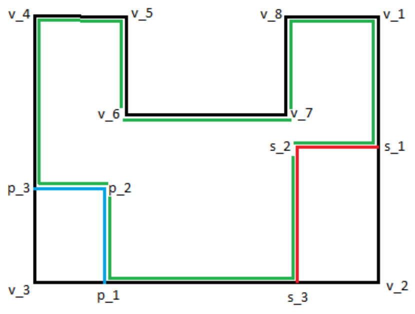


Figure 7.5: Example of edge intersected by two polylines

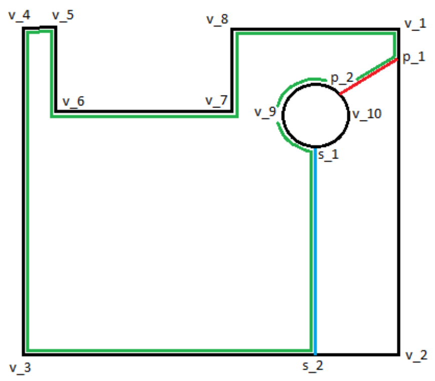


Figure 7.6: Example of polyline that starts from outer loop and ends on an inner loop

loop that was previously internal should be eliminated. Fig.7.6 shows an example of a polyline that starts from the outer loop and ends on the inner loop. In this case the new outer loop, drawn in green, has some edges of both outer and inner loops. The new boundary is made of just one outer loop.

. Once new loop is built, it is necessary to consider if and which internal loops are still belonging to the edge of the face. Fig.7.7 shows an example of a polyline that subdivides a face in two parts and separates the inner loops. As we can notice, the first inner loop, after the application of the algorithm, is outside the face and so is not any more considered as a loop. Instead, the second and third loops are still inside the face and are considered as inner loops.

. A face can have two or more connected components. Fig.7.8 shows an example of a couple of polylines that subdivide a face in three connected components. In this case the orientation of the polylines creates two connected components that are considered as the same face with two outer boundaries. Obviously it is possible to separate this face in two

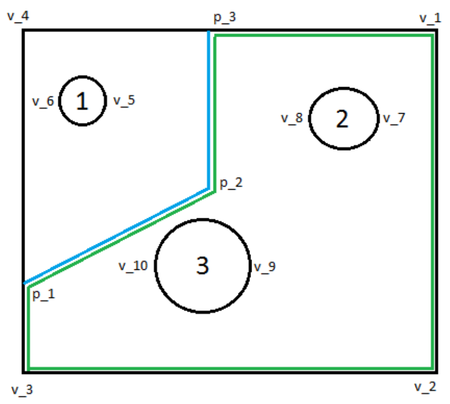


Figure 7.7: Example of inner loops inside (2,3) and outside (1) the new boundary

distinct faces.

This part of the algorithm gives the possibility to get loops making boundary of a face. This method is repeated for all the faces of the solids A and B involved in the boolean operation.

Creation of B-Rep results The last step is the creation of new faces starting from boundaries got in the previous step. Subsequently, new faces are added to the solid model R resulting by boolean operation. If a subtraction operation is performed, faces of the solid B got by the algorithm, are inverted in order to have consistent normal and a valid solid model. The result is a valid B-Rep model that can be considered the result of a boolean operation between solids A and B under a fixed tolerance.

In this chapter, we discussed in detail the main algorithm realized during the cooperation with Hyperlean. These algorithms give the possibility to increase the potentiality of an extremely useful software as LeanCost.

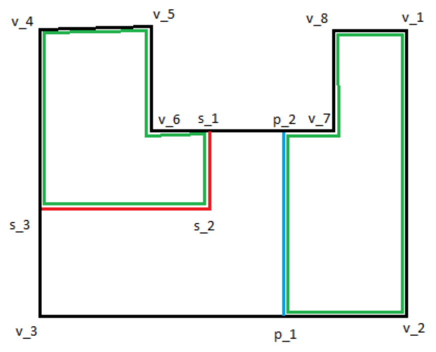


Figure 7.8: Example of face with two connected components

Conclusions and main results

The research project has been subdivided into academic and professional side. On the academic side we investigated many aspects of solid modeling, focusing on the B-Rep models and introduced a new paradigm which is able to integrate mesh and NURBS entities. To support this new modeling paradigm we proposed a suitable solid modeling system named “Extended Solid Modeling System”. On the other hand, the professional side of the project covered the development of algorithms in order to optimizate 3D geometry of solid objects and boolean operations between polygonal meshes improving the LeanCost software.

Academic performed activity

To formulate a complete proposal of an Extended Solid Modeling System, the academic side of this project was involved into two main goals:

- Study and design of a theory for a new paradigm of extended solid model representation that provide the basis for the definition of a new paradigm of an Extended Solid Modeling kernel.
- Development of tools to extend a classic solid modeling system, aimed at integrating the new primitives and the new paradigm.

Concerning the design of new paradigm of extended solid model representation, first of all we introduced the innovative paradigm “Extended B-Rep”

(EB-Rep) which is able to integrate the geometric entity “Mesh-Face” as part of a B-rep model.
Once the main characteristics of the EB-Rep were introduced, in order to realize this new system it was necessary to handle the interaction between classic B-Rep entities and meshes. In particular, the notion of continuity between Mesh-Face and NURBS entities had to be investigated, because it was necessary to define a new concept of continuity between smooth and discrete entities.
We formalized the definition of continuity between discrete and continuous entities: the Approximated Geometric (AG) continuity, conceptually introduced in [44]. In particular we analyzed the AG^0 and the AG^1 definition of continuity between Mesh-Faces and NURBS or analytic surfaces. Moreover we formalized the G^1-Almost-Everywhere Continuity, that is a different definition of G^1 continuity between Mesh-Faces and NURBS or analytic surfaces.
Then we provided a high-level overview of typical methods involved in solid modeling introducing their generalization to the Extended Solid Modeling System. In particular we focused on Boolean Operations, Cutting and Join Operation.
Concerning Boolean Operations, we investigated the NURBS-Mesh intersection. In this case the problem of intersection between two entities has been solved considering the AG^0 continuity and the result surfaces are a trimmed NURBS and trimmed Mesh-Face.
In Cutting Operation, similarly to the Boolean Operations, the surface-surface intersection between Mesh-Faces and NURBS surfaces is suitably managed, thus we used the same notions previously introduced.
Join operation has been analyzed handling the smooth joining between NURBS surfaces and Mesh-Faces according to the definitions of AG^1 and G^1-AE continuity.

The Development of tools to extend a classic solid modeling system required first of all to manage the efficient integration of the Mesh-Face prim-

itive into an existing Solid Modeling System based on a classical B-Rep paradigm.
At this aim, we proposed innovative approaches, suitable for both valence semi-regular and unstructured meshes.
Valence semi-regular quadrilateral and triangular meshes are represented by an EB-Rep with faces described by a low number of NURBS surfaces, without losing any information. The number of faces depends directly from the number of Extraordinary Vertices of the mesh.
We provided a standard and an improved version of the methods, called respectively *Quad Mesh Patching*(QMP) and *Quad Mesh T-Patching*(QMTP). The first one creates rectangular patches without T junctions while the second one allows T junctions, diminishing the number of patches necessary to represent the object, but losing the uniqueness of the solution, that depends from the order in which extraordinary vertices are considered. We implemented these methods in our OpenCascade platform in order to validate our proposal.
For unstructured meshes with arbitrary topology we studied and realized an innovative method that creates an EB-Rep with NURBS faces approximating the initial mesh. In particular, the proposed algorithm extended LSPIA algorithm introduced in [25] in order to obtain a Catmull-Clark surface approximating with good accuracy the original mesh. We tested this method by implementing it in our OpenCascade geometric kernel.
Furthermore, our innovative method is very interesting because of its multiple applications. It can be applied to obtain a valence semi-regular mesh from an unstructured one discretizing the Catmull-Clark surface, or it can be considered as a variant of Hoppe's surface reconstruction method, introduced in [32] and in [35], which works on an unstructured mesh obtained by a pre-processing of the original point cloud.

Concerning the second main goal of our academic project, we extended the most important tools for solid modeling to manipulate EB-Rep solids. In particular we extended the existing methods for Boolean Operations and Cutting Operation in order to manage EB-Reps. In both cases, the intersection curves are trimmed NURBS curves, if they delimit two NURBS surfaces, otherwise they are polylines that have AG^0 continuity with both the NURBS surface and the Mesh-Face. We produced examples of the proposed methods using our OpenCascade platform.
Then we analyzed in detail the Face-Join operation, introducing the 1-1 Face-Join operation, the 1-n Face-Join operation and finally the n-m Face-Join. In all these cases, the literature does not provides solutions for matching meshes and NURBS entities, thus we started from definitions of AG^0, AG^1 and G^1-AE continuity and realized innovative methods to join a NURBS surface and a Mesh-Face.
For the 1-1 Face-Join, we analyzed both the situations in which the NURBS is fixed and the Mesh-Face is fixed and introduced new methods that require the modification of the NURBS surface or the creation of a blending NURBS surface in order to close the gap between the two entities.
Methods concerning the 1-n Face-Join operation and the n-m Face-Join has been realized extending correspondent methods introduced for 1-1 Face-Join. proposed methods has been implemented and tested in our OpenCascade platform.

All the notions and methods introduced in our work allow us to formulate a complete proposal of an Extended Solid Modeling System and to provide the instruments to integrate the new primitives and the new paradigm into a classic B-Rep Solid Modeling System.

Professional performed activity

The professional side of our research activity has been subdivided in two main works that improved the LeanCost software: the development of algorithms for the optimization of the 3D geometry of solid objects and the study and realization of a method concerning boolean operations between polygonal meshes.

In the first part of our project we studied the Injection Molding Technology, in which a thermoplastic material is heated until it melts, then this melted plastic is forced into a steel mold, where it cools and solidifies. The main cost of Injection Molding is the creation of the mold. The more complicate is the mold, the more expensive is the production of the object.
In order to determine the complexity of realization of a 3D model using Injection Molding some fundamental characteristics of the object and of its mold has to be determined. In particular it is necessary to:

- determine if the object is moldable or not,
- estabilish which is the principal parting direction,
- find the best solution between all possible molding solutions.

We realized an innovative algorithm that solves these problems starting from the B-Rep representation of the model. In particular our algorithm firstly finds all the candidate extraction directions, computes Face Visibility for each direction and determines if the object is moldable. Then the Parting Direction and all molding solutions are computed.

The second part of our research activity involved the creation of an algorithm that gives the possibility to rebuild the boundaries associated with the faces of a B-Rep model obtained as the result of a boolean operation between two solids. More in details, two solids A and B, their associated mesh MA

and MB and the mesh MR result of Boolean Operation between MA and MB are given. Our algorithm obtains from MR the B-Rep solid R in which surfaces are parts of original surfaces of solids A and B.
The algorithm firstly extracts necessary information from Boolean operation result, then the new boundaries of the interested faces are created and finally the B-Rep model of the result is created.

Bibliography

[1] *http://github.com/VTREEM/Carve.*

[2] *https://www.blender.org/.*

[3] *http://www.cgal.org/.*

[4] *http://www.hyperlean.eu.*

[5] *http://www.opencascade.com/.*

[6] *Industrial automation systems and integration Product data representation and exchange, Overview and Fundamental Principles, International Standard, ISO TC184/SC4.*

[7] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Intl. J. Comput. Geom. Appl.*, 12:125–141, 2002.

[8] N. Amenta, S. Choi, and R. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2001.

[9] M. Antonelli. Metodi per la correzione delle superfici di suddivisione di Catmull-Clark intorno ai vertici straordinari, 2011.

[10] R. Barnhill, G. Farin, M. Jordan, and B. Piper. Surface/surface intersection. *Computer Aided Geometric Design*, 4:3–16, 1987.

[11] P. Benko, R. R. Martin, and T. Varady. Algorithms for reverse engineering boundary representation models. *Computer Aided Design*, 33:839–851, 2001.

[12] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva. State of the art in surface reconstruction from point clouds. 2014.

[13] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG*, 5:349–359, 1999.

[14] R. Bénière, G. Subsol, G. Gesquière, and F. L. Breton. A comprehensive process of reverse engineering from 3D meshes to CAD models. *Computer Aided Design*, 43:1382–1393, 2013.

[15] W. Boehm. Inserting new knots into b-spline curves. *Computer Aided Design*, 12:199–202, 1980.

[16] D. Bommes, T. Lempfer, and L. Kobbelt. Global structure optimization of quadrilateral meshes. *Eurographics 2011*, 30, 2011.

[17] D. Bommes, B. Levy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin. Quad-mesh generation and processing: a survey. *Eurographics*, 2012.

[18] J. Carr, R. Beatson, H. Cherrie, T. Mitchel, W. Fright, B. McCallum, and T. Evans. Reconstruction and representation of 3D objects with radial basis functions. *SIGGRAPH*, pages 67–76, 2001.

[19] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, 1978.

[20] K. Chan and C. Chen. 3D shape engineering and design parameterization. *Computer Aided Design*, 5:681–692, 2011.

[21] X. Che, X. Liang, and Q. Li. G1 continuity conditions of adjacent NURBS surfaces. *Computer Aided Geometric Design*, 22:285–298, 2005.

[22] Z. Chen, X. Luo, L. Tan, B. Ye, and J. Chen. Progressive interpolation based on Catmull-Clark subdivision surfaces. *Computer Graphics Forum*, 27:1823–1827, 2008.

[23] H. Chiyokura. *Solid Modeling with DESIGNBASE*. 1988.

[24] C. K. Chui, M. J. Lai, and J. Lian. Algorithms for G1 connection of multiple parametric bicubic NURBS surfaces. *Numerical Algorithms*, 30:285–313, 2000.

[25] C. Deng and H. Lin. Progressive and iterative approximation for least squares B-spline curve and surface fitting. *Computer-Aided Design*, 47:32–44, 2014.

[26] A. Duster, J. Parvizian, and E. Rank. Finite cell method: h- and p- extension for embedded domain methods in solid mechanics. *Computational Mechanics*, 41:122–133, 2007.

[27] A. Duster, J. Parvizian, Z. Yang, and E. Rank. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 197:3768–3782, 2010.

[28] G. Farin. *Curves and surfaces for CAGD: a practical guide.* Morgan Kaufmann, 2002.

[29] G. Farin, J. Hoschek, and M. S. Kim. Handbook of Computer Aided Geometric Design. 2002.

[30] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[31] M. E. Hohmeyer. *Robust and Efficient Surface Intersection for Solid Modeling.* PhD thesis, EECS Department, University of California, Berkeley, May 1992.

[32] H. Hoppe. Surface reconstruction from unorganized points. 1994.

[33] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26:71–78, 1992.

[34] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. pages 19–26, 1993.

[35] H. Hoppe and M. Eck. Automatic reconstruction of b-spline surfaces of arbitrary topological type. *SIGGRAPH*, pages 325–334, 1996.

[36] J. Huang and C. Menq. Automatic CAD model reconstruction from multiple point clouds for reverse engineering. *Transactions of the ASME*, 2:160–170, 2002.

[37] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[38] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. on Graphics*, 34:189:1, 189:15, 2015.

[39] Y. Kineri, M. Wang, H. Lin, and T. Maekawa. B-spline surface fitting by iterative geometric interpolation/approximation algorithms. *Computer Aided Design*, 44:697–708, 2012.

[40] R. Kolluri, J. Shewchuk, and J. OBrien. Spectral surface reconstruction from noisy point clouds. *SGP*, pages 11–21, 2004.

[41] H. Lin, W. Chen, and H. Bao. Adaptive patch-based mesh fitting for reverse engineering. *Compute-Aided Design*, 39:1134–1142, 2007.

[42] T. Maekawa, Y. Matsumoto, and K. Namiki. Interpolation by geometric algorithm. *Elsevier Science*, 2007.

[43] M. Mantyla. Introduction to solid modeling. 1988.

[44] P.Besl. Hybrid modeling for manufacturing using NURBS, polygons, and 3D scanner data. *IEEE*, 5, 1998.

[45] A. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12:437–464, 1980.

[46] D. Schillinger and M. Ruess. The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models.

[47] M. Schweingruber-Straten. Generierung von Oberflächennetzen nach der Gebietsteilungstechnik, 1999.

[48] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *SIGGRAPH 1998*, pages 395–404, 1998.

[49] T. Takamura, M. Ohta, H. Toriya, and H. Chiyokura. A method to convert a gregory patch and a rational boundary gregory patch to a rational bèzier patch and its applications. *Proceeding Computer Grapthics International '90*, 1990.

[50] W. Tiller and L. Piegl. *The NURBS book*. Springer Verlag, 1997.