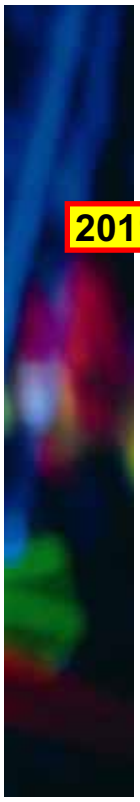
ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

ARTS ET MÉTIERS ParisTech

Università degli Studi di Genova



~~2010-ENAM-XXXX~~

(provisional version)

2011-ENAM-0017

PhD T H E S I S in cotutelle

to obtain the degrees of

Docteur ~~de~~ des

~~l'~~Arts et Métiers ParisTech

Spécialité "~~Sciences des Métiers de l'Ingénieur~~ Conception"

École doctorale n° 432

and

Dottore di Ricerca della

Università degli Studi di Genova

Specialità "Meccanica e Costruzione delle Macchine"

presented and defended publicly by

Ruding LOU

June 21st, 2011

Modification of semantically enriched FE mesh models~~:~~

Application to the fast prototyping of ~~alternate~~ alternative solutions in the context of industrial maintenance

Directors of thesis: **Philippe VÉRON** and **Bianca FALCIDIENO**

Co-Directors of thesis: **Jean-Philippe PERNOT** and **Franca GIANNINI**

Industrial responsibles: **Alexei MIKCHEVITCH** and **Raphaël MARC**

Jury

Mme. Dominique BECHMANN, Professor, Université de Strasbourg	REVIEWER
Mme. Stefanie HAHMANN, Professor, INP-Grenoble	EXAMINER
M. Umberto CUGINI, Professor, Politecnico di Milano	REVIEWER
M. Rinaldo C. MICHELINI, Professor, Università di Genova	EXAMINER
M. Philippe VÉRON, Professor, Arts et Métiers ParisTech	PhD co-supervisor
M. Jean-Philippe PERNOT, Associate Professor, Arts et Métiers ParisTech	PhD co-supervisor
~~**Mme. Bianca FALCIDIENO**, Research Director, CNR-IMATI.Ge~~	~~PhD co-supervisor~~
Mme. Franca GIANNINI, Senior Researcher, CNR-IMATI.Ge	PhD co-supervisor
M. Alexei MIKCHEVITCH, PhD Engineer, EDF Division R&D	Industrial partner
M. Raphaël MARC, ~~PhD Engineer~~ Research Engineer, EDF Division R&D	Industrial partner

THESIS

Arts et Métiers ParisTech - Centre d'Aix en Provence, LSIS-UMR CNRS 6186 France
Università degli Studi di Genova, CNR-IMATI.Ge, Italia

Acknowledgements

At this place I would like to acknowledge all the people who have helped me with my PhD. At the first place the greatest thanks go to my thesis directors Prof. Philippe VERON and Dr. Bianca FALCIDIENO for their guidance and support.

I would like to express my deepest recognition to Dr. Jean-Philippe PERNOT, my thesis co-supervisor in France. Since my master thesis he has brought me into the computer graphics research work and made me interested by it. Thanks to him I have learned much advanced knowledge on the topics and all the surrounding fundamental science.

I have to appreciate the cares from Dr. Franca GIANNINI my thesis co-supervisor in Italy, both for the scientific quality of her guidance and the kind reception she did during my first residence in Italy.

I would like to thank all the members of the jury, Prof. Dominique BECHMANN, Prof. Ste-fanie HAHMANN, Prof. Umberto CUGINI and Prof. M. Rinaldo C. MICHELINI, for having accepted to evaluate my thesis.

Stefanie

MIKCHEVITCH Raphaël

Thanks to Dr. Alexei MIKCHEVITH and Dr. Raphal MARC, my advisors of industrial partner from EDF, I have discovered for the first time the engineering in the energy industry in France that has given much application meaning to my thesis.

I also have to thank my ex-colleague Dr. Minica PANCHETTI, who has helped me in mesh modelling, as well as all the other members in the lab. LSIS in France. I will not forget all my colleagues in IMATI as well as their friendly reception during my research work in Italy.

At last I would like to thank my parents who have taken care on me and pushed me to go further on my studies. I would also like to express my appreciations to my fiance for her loves since 10 years and forever.

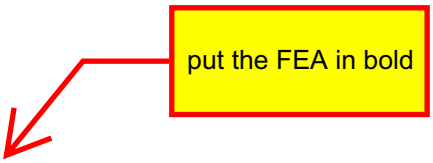
Abstract

Behaviour analysis is largely performed on the virtual model of the product before its physical manufacturing. The move from the reality to the digital world is gainful since it avoids the high costs in terms of money and time spent on intermediate manufacturing required for performing simulations on real products. Anyhow, the process could be further optimised especially during the product behaviour optimisation phase. This process involves repetition of four main processing steps: CAD design and preparing for meshing, mesh creation, enrichment of the model with physical semantics and Finite Element Analysis (FEA). The product behaviour analysis is performed on the first design solution as well as on the numerous successive product optimisation loops. Each design solution evaluation necessitates the same time as required for the first product design and it is particularly crucial in the context of maintenance and lifecycle assessment. This thesis proposes a new framework for CAD-less product optimisation through FEA which reduces the mesh preparation and FEA semantics enrichment activities. More concretely, the idea is to directly operate on the firstly created FE mesh, enriched with physical/geometric semantics, to perform the product modifications required to achieve its optimised version. In order to accomplish the proposed CAD-less FE analysis framework, modification operators acting on both the mesh and the associated semantics need to be devised. In this thesis, the underlying concepts and the devised components for the development of such operators are discussed. A high-level operator specification is proposed according to a modular structure that allows an easy realisation of different mesh modification operators. Finally, four instances of this high-level operator are described: merging, cracking, drilling and filleting. These operators are prototyped and validated on academic and industrial FE mesh models, thus clearly showing the feasibility of our approach.

Contents

List of Figures

Introduction



The Finite Element Analysis (FEA) plays a key role for evaluating the multiple solutions in the context of product design and maintenance. This is due to the fact that FEA helps to reduce the time development and/or maintenance times while saving money. It reduces material costs by testing for safety and behaviour with less material or alternative materials and it reduces physical prototyping costs by proving and improving the product design before it is produced. It avoids long-time physical prototyping of the various envisaged product optimisation solutions. All these advantages are very valuable for competitive engineering and maintenance purposes. Since FEA is widely used in numerous industrial companies, it is crucial to think whether the FEA workflow can be optimized and improved to save additional time as well as money. Nowadays, the mainstream methodology for product behaviour analysis and conceptual solution assessment using FEA relies on the following steps: 1) Computer-Aided Design (**CAD**) modelling, meshing and preparing the CAD model for specific meshing, 2) Finite Element (**FE**) mesh meshing, 3) Physical semantics modelling, 4) Finite Element simulation and result analysis. These four stages constitute the loop for a product design solution evaluation. According to the obtained results of the FE simulation, experts may start another optimisation loop for another design solution or improve the FE modelling; we refer to this process cycling as optimisation loop. Each optimisation loop for each product design solution necessitates repeating all the above listed four stages from CAD until simulation.

However, in the context of current maintenance and lifecycle problem analysis, the product already exists, the CAD models are not necessarily available and the product behaviour has to be studied and improved during its exploitation. Companies exploiting industrial complex installations are currently submitted to

various constraints crucial from a production point of view. They can be relative to the time and cost of the production process stops, to the efficiency of maintenance solutions, to production safety criteria, etc. For example, in the field of power production, it is critical to identify the problem source and to provide the appropriate solution while taking care of the triplet: Time, Quality and Cost. As a reference, the total cost of one day of stop of a nuclear power station represents several hundreds of thousands of Euros. In the context of power production equipment maintenance, it seems quite clear that the reduction of the time of the projects carried out by engineering departments (fast operational study, solution optimisation for maintenance) may lead to important research perspectives in terms of fast numerical prototyping and solution assessment methods during the operational study. This reduction concerns the time spent notably for various numerical simulations (e.g. mechanical resistance assessment, vibration analysis, lifecycle contra-expertise). We found that for each optimisation loop we spend a lot of time on preparing the FEA model: 1) The CAD modelling necessitates modifying the previous CAD model or creating completely a new one if it is not available; 2) the mesh modelling stage needs to make the CAD model meshable and to specify different characteristics on the mesh: different sizes of mesh elements, non-manifold meshing, (i.e. double mesh entities, disjoint entities, $\cdots$), etc.; 3) to simplify defining different physical semantics on the mesh model, different groups of mesh elements should be created before. The first result of simulation allows tuning/certifying the model, in case of failure it is necessary to come back to the CAD/meshing/FE model creation stages for modifying the simulation model. In the context of product maintenance, the structural optimisation modification concerns locally around the problem zone. For saving time the interesting way for fast evaluating the solution by FEA is to directly prototype the modification on the existing FEA mesh model previously validated.

In this thesis, the definition of a new prototyping method has been investigated: the proposed CAD-less fast prototyping approach avoids going back to the CAD models during the fast numerical solution assessment. Actually, the idea is to enable direct modification of complex meshes enriched with semantics for fast and accurate FE simulation of production installation behaviour during the operational studies. This approach should avoid multiple and time-consuming

iterative updating of the CAD models, as well as the tedious re-meshing steps of potentially complex shapes. Not only the newly defined modification operators have to take into account multiple geometric aspects and constraints, but they also have to consider semantic data associated to the FE meshes: groups of FE entities must be maintained during various mesh modifications. The FE groups can contain nodes, edges, faces and 3D elements (tetrahedra, hexahedrons, etc.) or the mix of different topology FE elements. These semantic data enrich the FE mesh models with information that are classically relative to the diverse parameters of geometric and mechanical nature (materials, FE modelling type, boundary conditions, external loads, etc.) required for FEA. The proposed approach is particularly efficient in the case of fast prototyping of local structural modifications on a given production installation since it does not worth to redo the entire FEA semantics enriched mesh for making locally small modifications. It can also be applied during the product preliminary design phases where many alternative solutions often have to be tested and simulated. This thesis is organised in two parts. The first part introduces the models, methods and tools classically adapted for FE Analysis (chapter 1) and details the way enriched meshes can be modified using today's' available tools (chapter 2). In the second part, the proposed CAD-less approach is presented. Chapter 3 describes the aspects of a model that have to be taken into account in the generic approach. The basic methods and tools for modifying the meshed geometry are introduced in the chapter 4 whereas the basic methods for manipulating semantics in a mesh are presented in the chapter 5. Finally, the chapter 6 describes how a generic CAD-less operator is built from newly introduced basic methods and tools. It also details different instances of such a operator. The last chapter ends the manuscript while concluding and opening perspectives to this work.

PART I Hight-level manipulations of Finite Elements meshes : state-of-the-art

In the first part, the general context of this PhD thesis is introduced. The context includes scientific and industrial background. This thesis is located in the domain of product design and maintenance studies aided by computer threfore different product design approaches are presented at first. Then the industrial engineering context is shown as well as some concret industrial examples in maintenance based on EDF's (Électricité De France) needs are given. The typical problem from industrial point of view today is that the model preparing for FEA is very heavy in the sense that for FE model improvement as well as new design product assessing the classical FEA loop should be repeated.

In order to speed up the FEA model preparation process, this thesis proposes to manipulate directly the FE semanticlly enriched meshes without going back to the CAD model. Therefore a full state of art on manipulations of FEA meshes are then analysised. Various criteria are listed in order to analyse whether the research works could response the industry needs.

At the end, we propose our method allowing modifying directly the FE semantics enriched meshes taking into account different industrial constraints as mesh quality, FE group presence, etc.

Chapter 1

Models, methods and tools for industrial studies

Computer-Aided Engineering (**CAE**) is wildly spread in the world along with the development of computer science. CAE is the broad usage of computer software to assist in engineering tasks along the product life cycle (**PLC**). This includes Computer-Aided Design (**CAD**), Computer-Aided Analysis (**CAA**), Computer-Integrated Manufacturing (**CIM**), Computer-Aided Manufacturing (**CAM**), Material Requirements Planning (**MRP**), and Computer-Aided Planning (**CAP**). Along these different phases the physical solid to be studied is represented by a **numerical geometric model**.

Today in most industries Computer-Aided Engineering is normal practice. Production is driven by Computer-Aided Design, Analysis, and Manufacturing. All the data relative to the product development process are stored and managed by Product Data Management (**PDM**) systems which may contain all the models used at different steps. For processing a so-called Digital Mock-Up (**DMU**) various tools and methods are mandatory. In the first part of this chapter, an overview of the most used methods for geometric modelling are presented. In the second part, the manipulation of the DMU is introduced and notably the techniques relative to the adaptation of the DMU for FEA, which is the core of the thesis. Different industrial highlighted problems are illustrated from examples in maintenance numerical studies of power production equipment.

1.1 Geometric models along the product lifecycle

The **geometric modelling** concerns the methods and algorithms for describing and manipulating digital shapes [2, 92]. A model is a mathematical abstraction of a physical object or phenomenon and has to be converted into a representation to be computationally treated. In product development, geometric models are created during the design phases for describing the shapes of the product at the beginning of the PLC development using CAD tools. Then, according to the various needs (simulation, manufacturing, etc.) different models are needed. Generally, the geometric modelling consists in creation, exchange, visualisation, animation, interrogation, and annotation of digital models of physical objects. Different approaches of geometric modelling are presented in the following subsections.

1.1.1 Early representation schemes

Different early model representations are presented in [2]. Approaches of geometric modelling have begun before the rise of computers and with pencil and paper. Since the advent of computers, the approaches of geometric modelling have changed a lot. **Engineering drawings** were the earliest attempts to model objects. Computers were not involved and they were intended as a means of communication among humans. The engineering drawing often had errors but humans were able to use common sense to end up with correct result. There was no formal definition of such drawings as a representation scheme. The basic idea was to represent objects by a collection of planar projections.

Wireframe representations were the first representation schemes adopted for three-dimensional linear polyhedrons in computer-aided tools. It is a natural approach, the idea being to represent objects using only their edges. But wireframe representations are ambiguous, the well-known example that consists of 16 vertices and 32 edges is shown in figure 1.1.a. It represents a solid and each of the quadrilaterals (some of them are squares) defines a face of the solid. The inner cube can represent a hole but it is not defined the direction of the opening of the

cube. Therefore, it could be one of the three possibilities (fig.1.1.c - fig.1.1.d) for this opening. Some works have been done for identifying the solid from wire-frame models [118, 119].

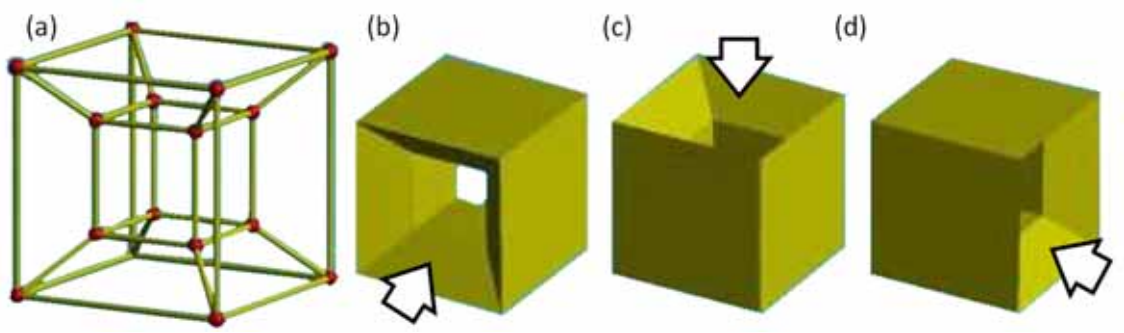


Figure 1.1: Wireframe model representation and possible solids can be deduced from it

Many early commercial modelling systems have used wireframe representations. However, wireframe models are ambiguous and are not anymore used for representing the object in today's applications but for visualisation purposes. Many systems support a wireframe display mode because it is efficient (i.e. only vertices and edges are displayed and processed). For example, wireframe visualisation can be used for preview purpose. Rendering a complex model or an animation sequence could be very time-consuming if all objects have to be rendered. If wireframe models (usually including its face information) are available, one can easily obtain a general feeling of the final result without waiting for minutes or even hours before spotting a design flaw. The edges in a wireframe model do not have necessarily to be line segments. They can be curve segments and in this case the edge table will be more complicated since in addition to the two endpoints a description of the joining curve segment (e.g. an equation) is required.

Faceted representation is a simple solution which circumvents the major limits of the wireframe representation by adding faces. This representation is ambiguous in collision detection for the case where two objects are intersecting and their faceted representations are not intersecting. There is a difference between a modelling system using a faceted representation and one using a faceted display.

The second one means that objects are displayed by linear polyhedra even though the system may maintain an exact analytic representation of objects internally.

Parameterised primitive instancing [92] scheme is based on the notion of families of objects, each member of a family being distinguishable from the other by few parameters. Every scheme has primitives, which must be instantiated to construct structured representations. Primitives may be low level, e.g. the points used to represent polygons in the previous section. But they may also be high level. For example, many modelling system have solid primitives such as blocks and cylinders. Each object family is called a generic primitive, and individual objects within a family are called primitive instances. Each primitive is represented via tuples of the form

(type code, parameter 1 , . . . , parameter k)

where the parameters are either reals or integers. For example, a solid block aligned with the principal axes can be represented by the 4-tuples ('block', Length, Width, Height), where the last three parameters are real numbers that define the dimensions of the block, as shown in figure 1.2.a. For another example, a family of bolts is a generic primitive (fig.1.2.b) that can be represented by the 3-tuples ('bolt', H, D) and a single bolt specified by a particular set of parameters is a primitive instance.

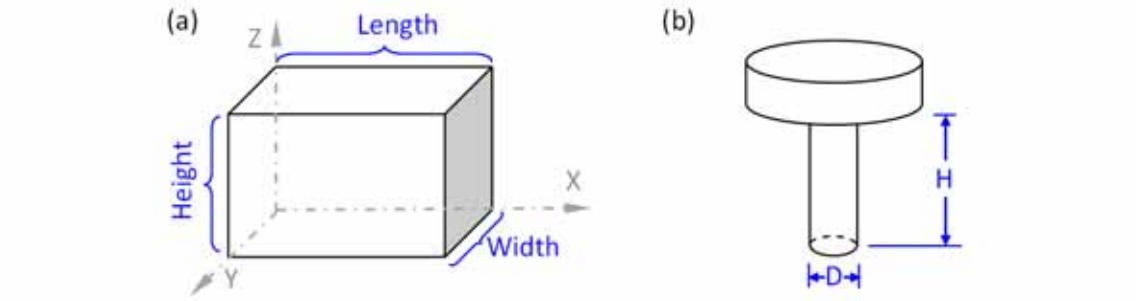


Figure 1.2: Parameterised primitive instancing schema examples

As for the example of the bolt, not all the dimensions are needed as parameters, only those that are variable. The representation is unambiguous and may be unique. It is certainly very compact. Pure primitive instancing schemes, which

have no structured representations, are not very attractive. It is like a language that has words but no sentences. They tend to have a small domain. In addition, each primitive type requires special-case algorithms for evaluating its properties. Primitive instancing is important primarily in the context of other schemes that not only instantiate primitives but also combine them into higher-level structures.

1.1.2 Decomposition schema

Decomposition representation consists in two types: "objet-based" or "space-based" [2]. The object-based versions present a subdivision of the object itself whereas the space-based versions subdivide the whole into elementary spaces then mark those that belong to the object.

The cell decomposition is an object-based decomposition representation. The model is broken into primitive pieces (cells) typically triangles in two-dimensional case or tetrahedral pieces in three dimensional case. The primitive cells usually have relatively simple definition. The mesh presented more in detail in sub-section 1.1.7 (p.16) is belongs to a kind of cell decomposition.

The space-based decomposition schema consists in partitioning the space into regions called cells, and to enumerate those cells which are filled with material and therefore constitute the object being represented [92]. Decompositions into regular, fixed-size cells are called spatial occupancy enumerations. In 2D, the primitive square cells are sometimes called pixels, and in 3D, cubical cells are called voxels. Regular decompositions usually represent only "staircase" approximations of the desired objects. For reasonable accuracy, decompositions become very large, to overcome this drawback, hierarchical decompositions with cells of varying sizes have been introduced. In hierarchical decomposition, small cells are used only where required for an accurate approximation. 2D quadtrees are used extensively in image processing.

The figure 1.3.a shows a simple 2D polygon with sides parallel to the principal axes X-Y. The 2D spatial decomposition representation for this object is shown in figure 1.3.b. In each spatial tetrad-decomposition, quadrants are numbered from 0 to 3 in clockwise order, as shown at the right top. The 2D spatial decomposition can be represented into a quadtree (fig.1.3.c) that has three types

of nodes. Grey nodes correspond to cells that are neither completely full nor completely empty. Such cells must be subdivided. Black nodes are full and white nodes are empty. The root node of the quadtree corresponds to the entire space, i.e., to a square box that encloses the object. The quadtree may be constructed by the following (conceptual) procedure. If a cell is full or empty, mark it black or white, respectively; otherwise mark it grey, subdivide it, and recurse.

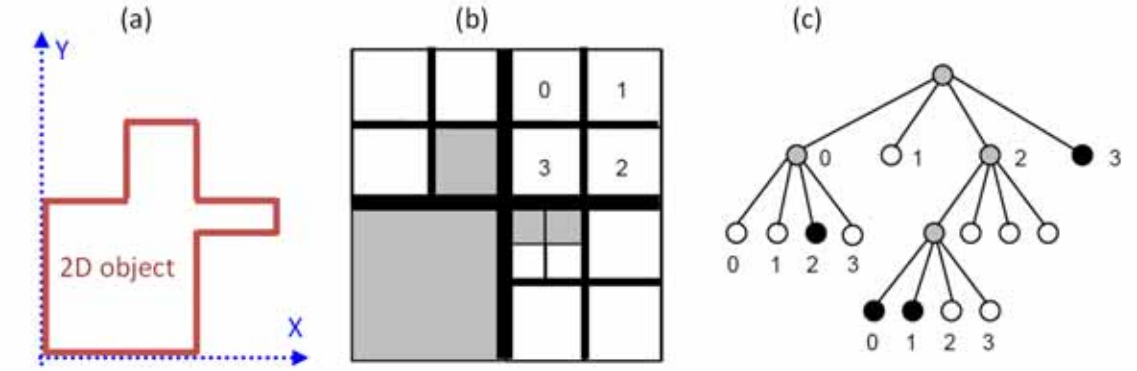


Figure 1.3: A 2D object represented by 2D spatial decomposition

1.1.3 Constructive Solid Geometry (CSG)

The Constructive Solid Geometry (**CSG**) modelling strategy implicitly represents a solid as an algebraic expression defined by a sequence of operations for constructing the object. The operation sequence is typically stored as a tree [5]. The modelling operators are regularised Boolean operations as union, intersection and difference, and rigid-body motions. The operands are primitive solids, classically block, sphere, cylinder, cone and torus, instantiated to specific dimensions.

In figure 1.4, the binary tree with 4 levels is used to build a 3D object from a set of 3D solid primitives like cube, sphere and cylinder. The final and intermediate objects in the figure are constructed by applying Boolean operations on the lower level objects that could be either primitive objects or intermediate objects. For example the left intermediate object at level L_1 is defined from the intersection between two primitives at level L_2, a cube and a sphere. The right object at level $L1$ is a union of a primitive cylinder and an intermediate object created by

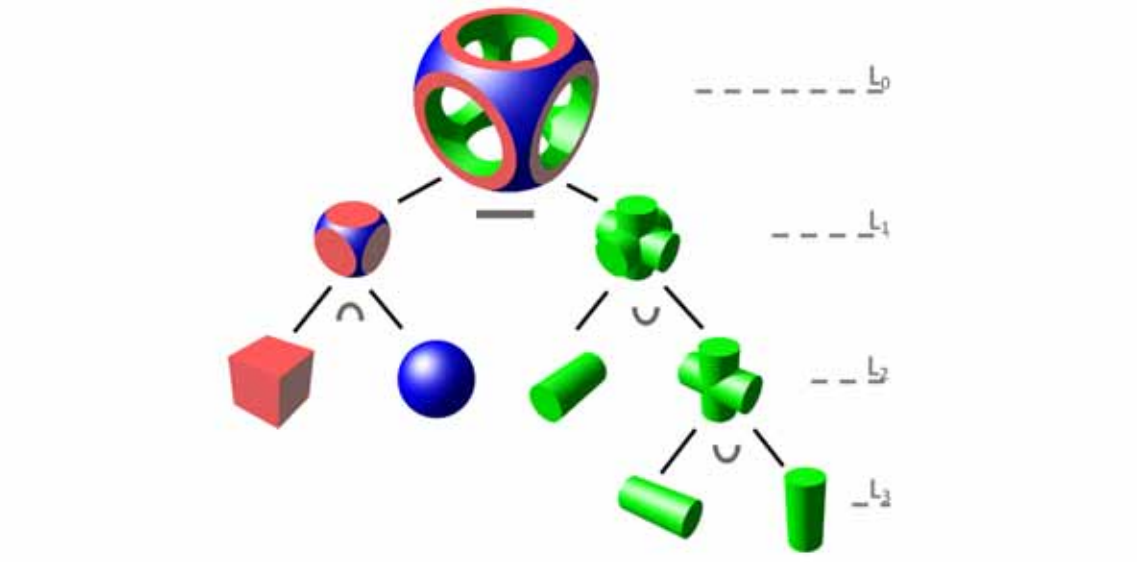


Figure 1.4: CSG in 3D modelling

union between two cylinders at level L_3. Using the general CSG operations gets a representation that is (1) unambiguous, (2) not unique, (3) very concise and (4) easy to create.

The CSG representations are compact, but they are not unique (different construction trees may leads to the same shape). The final object is not explicit represented, e.g. for wireframe visualisation the edges are not directly available but should be evaluated when needed. There are difficulties in treating free form surfaces since they are based on half-space representations. These limits somehow are the motivation why boundary representations are standard in current CAD systems.

1.1.4 Boundary representation (B-Rep)

Boundary representation **B-Rep** explicitly represents the solid through its boundaries. Topologically, the surface is a set of vertices, edges and faces, where the adjacencies are represented. Geometrically, a face is a (well-behaved) subset of a surface. More concretely, B-Rep is based on the decoupling of the topological elements constituting the object boundary with the associated geometric information. The basic topological elements are faces, edges and vertices, possiby

grouped according to their connection in loops and shells. The geometry is given by the mathematical equation of the surface corresponding to the face or the curve corresponding to the edge and to the 3D coordinates of the point on which the vertex is located. The surface could be a parametric surface or patch, an implicit algebraic surface, or a procedurally represented surface.

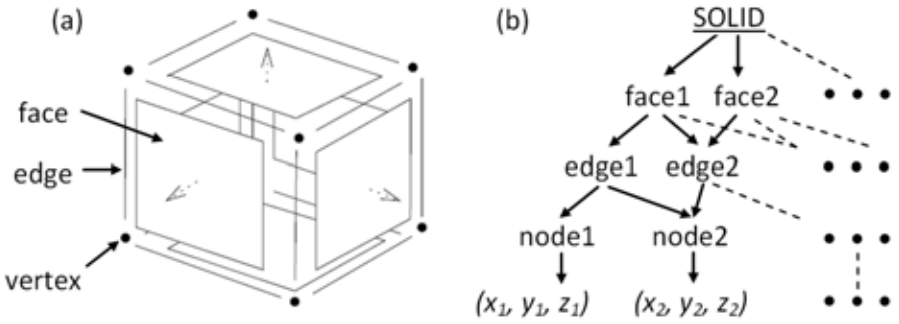


Figure 1.5: A Boundary Representation used for object modelling and corresponding graph

Since humains see the surface of solids, it is natural to represent them via their boundaries. Mathematically, this is justified because in the special case of closed and bounded solids in R^n the boundary of a solid uniquely defines that solid. In B-Rep schema [17], a solid is represented by decomposition of its boundary in terms of faces, edges and vertices. A distinction is drawn between the topological entities (vertex, edge and face), related to each other by incidence and adjacency relationships, and the geometric location and shape of these entities. For example when polyhedra are represented, the faces are polygons described geometrically by a plane equation, edges represent the polygon boundaries and their geometry correspond to a line equation, whereas vertices correspond to the limits of the edges having associated a position in the space (i.e. the vertex geometry). Geometrically, B-rep entities are not allowed to intersect anywhere except in edges and vertices that are explicitly represented in the topology data structure or can be derived trough adjacency relationships explicitely represented.

One example of B-Rep is shown for a regular cube-like model (fig.1.5.a). The B-Rep contains 6 faces, 12 edges and 8 vertices that are partially shown in a hierarchic graph (fig.1.5.b) representing the connectivities.

Boundary representation has also been extended to allow special model types called **non-manifold models**. Normal or regular solids found in nature have the property that, at every point on the boundary, a small enough sphere around the point is divided into two pieces, one inside and one outside the object. Non-manifold models break this rule. An important sub-class of non-manifold models concerns mixed-dimentional models as 3D sub-components related to sheet sub-components used to represent thin parts may be idealised from mechanical point of view.

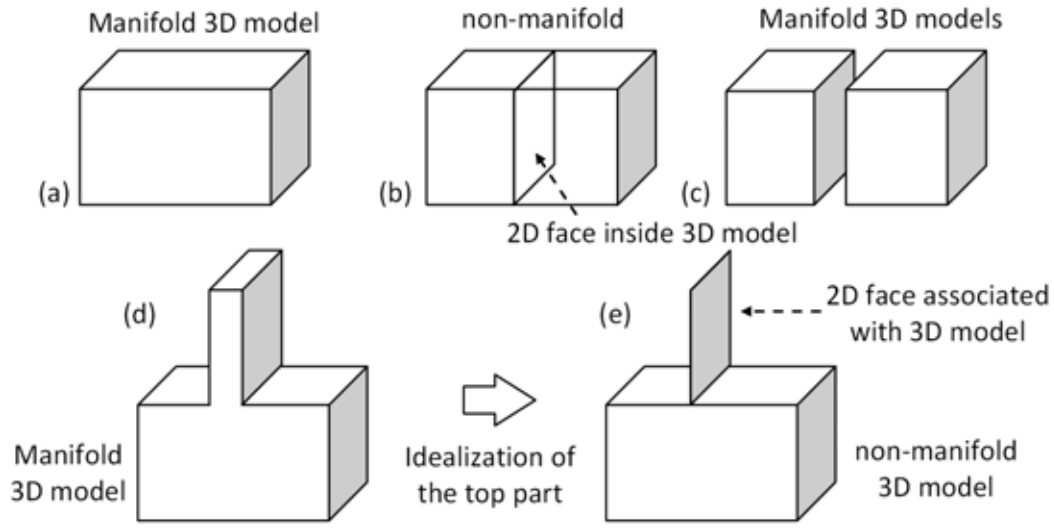


Figure 1.6: Non-manifold B-Rep modelling of an object

The figure 1.6.a shows a manifold 3D model represented by its 6 rectangle faces, 12 edges and 8 vertices. Once a surface is inserted into the model (fig.1.6.b) this model becomes non-manifold. But if the model is completely split into two sub parts (fig.1.6.c) the two sub-models become manifold. The non-manifold B-Rep is used in industry (for example, at EDF-R&D) for advanced modelling and FE analysis aims. In figure 1.6.d manifold model is composed by an upper part and a lower part. For certain reason (ex. speeding up the computation) some parts of the model could be idealised/simplified into 2D part. In the example shown (fig.1.6.d) the idealisation of the top part produces a non-manifold model (fig.1.6.e). Non-manifold models like the one presented on figure 1.6.b can also be used to split the model in two sub-partitions that are used during the FE

simulation to assign two materials, for example.

1.1.5 Sweep representation

The sweeping operation can be considered as another constructive method [108]. Sweep representations correspond naturally to the way many mechanical parts are manufactured. The basic idea of this schema is to "sweep" one set A along another B. It consists in moving a set through space to trace or sweep out a volume (a solid) that may be represented by the moving set and its trajectory. Such a representation is important in the context of applications such as modelling of tube-like systems, detecting the material removed from a cutter as it moves along a specified trajectory, computing dynamic interference of two solids undergoing relative motion, motion planning, and even in computer graphics applications such as tracing the motions of a brush moved on a canvas. Most commercial CAD systems provide limited functionalities for constructing swept solids mostly in the form of a two dimensional cross section moving on a space trajectory transversal to the section.

For example, the created object (fig.1.7.b) is obtained by sweeping the section along the guide (fig.1.7.a). The section is a 2D object that is repeated along the guide curve for generating a volume model.

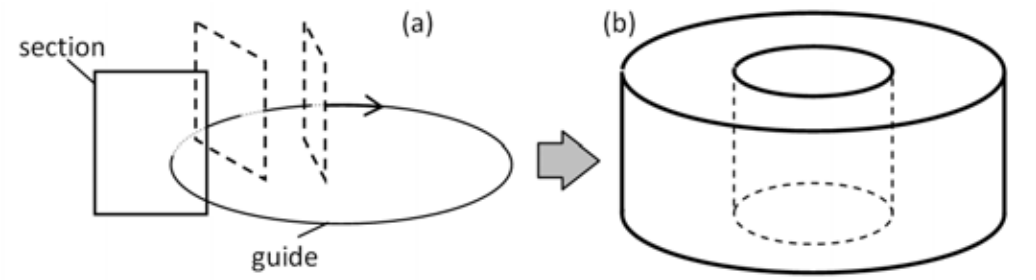


Figure 1.7: Sweep a section along a guide

1.1.6 Implicit and Parametric representations

There are two main techniques for representing surfaces in geometric modelling and computer graphics: implicit and parametric representations [36]. Both meth-

ods have distinct and complementary advantages. In the case of implicit surfaces, it is straightforward to decide whether a given point in space is or is not on the surface. For a parametric surface, on the other hand, it is easy to generate points that lie on the surface.

1.1.6.1 Implicit method

Implicit method represents solid as the set of points where an implicit global function takes on a certain value. Therefore, every algebraic surface in affine 3D-space is determined by an implicit equation:

$$f(x, y, z) = 0 \tag{1.1}$$

where $f(x, y, z)$ is a function in the unknowns x, y and z. The surface consists of all points (x; y; z) that satisfy this equation. In solid modelling, real coordinates are considered. Since the given equation may have more solutions that we want therefore the implicit method could represent for example many quadratic surfaces, such as sphere, ellipsoid or torus.

1.1.6.2 Parametric method

Parametric representations typically define a surface as a set of points P (s, t) such as:

$$P(s, t) = (X(s, t), Y(s, t), Z(s, t)) \tag{1.2}$$

For specific values of s and t, these functions (X, Y and Z) assign the coordinates of a surface point in Cartesian space. Parametric methods were motivated by properties of coordinate system independence, single-valued functions, ease of handling vertical slopes, and efficient evaluation of points on the surface. The last is critical for image rendering therefore in computer graphics. Most used parametric surfaces are Bezier surface, non-uniform rational B-splines or NURBS and so on, which are handled by current CAD systems. The parametric method has some limits. For example, it is impossible to get multiple values of $f(x, y)$ for a given x; hence circles and ellipses must be represented with multiple curve segments. Furthermore, such representation is not rotationally invariant and

describing curves with vertical tangents is difficult, because a slope of infinity is difficult to represent.

1.1.7 Meshes

According to [50], a mesh M is a geometric discretisation of a domain Ω that consists of:

- a collection of mesh entities M_i^d (ith entity of dimension d) of controlled size and distribution;
- topological relationships or adjacencies forming the graph of the mesh.

The mesh M covers Ω without neither overlap nor hole. A mesh is decomposition of a domain Ω with elements so that the computation over the approximated geometry is easier. In computer graphics tesselletions, i.e. discrete models are normally referred to meshes whereas in engineering they have a more specific interpretation, i.e. a tessellation with specific characteristics. Meshes are defined on top of basic elements:

- nodes M_i^0 that are topological entities of dimension 0,
- edges M_i^1 that are topological entities of dimension 1,
- faces M_i^2 that are topological entities of dimension 2,
- volumic elements M_i^3 that are topological entities of dimension 3.

Mesh entities have simple shapes. They are mainly segments in 1D, triangles and quadrangles in 2D and tetrahedra, hexahedra, pyramids and prisms in 3D.

A mesh M is composed of a collection of mesh entities together with their adjacencies. Any mesh entity bounds and/or is bounded by entities of higher and/or lower dimension. This adjacency information represents the graph of a mesh. A mesh is used to display a model with computer, , for instance, in virtual reality environment, since computers displays only triangles. A mesh is also used in FEA in order to simulate physical behaviour of an object.

A triangle mesh is shown in figure 1.8.a. Being defined this surface mesh, the volume mesh could be generated in different way. If this surface mesh is closing a space, the whole volume can be filled in (fig.1.8.b). In case, the user want to model the thickness (fig.1.8.c), tetrahedra can be put in a bandwidth around the surface mesh. Note that for FEA, the thickness can also be directly taken into account as a geometric attribute of the surface mesh modelling a thin object. The main technique for mesh generation are octree decomposition [102], advancing front method [70], Delaunay criterion [34] and so on. The mesh techniques are detailed in the next section .

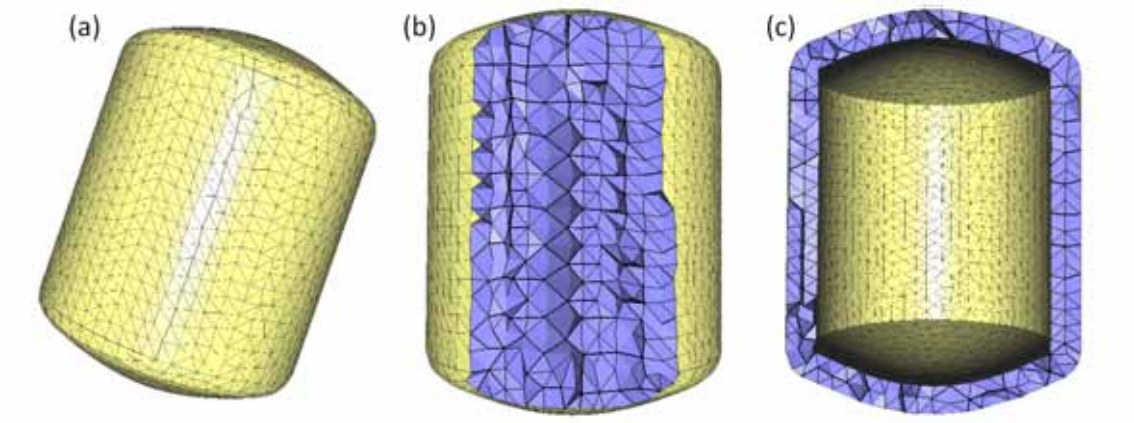


Figure 1.8: Examples of surface mesh (a) filled in completely (b) or partially (c)

1.1.8 Subdivision surfaces

A subdivision surface [125], in the field of 3D computer graphics, is a method for representing a smooth surface via the specification of a coarser piecewise linear polygon mesh. The smooth surface can be calculated from the coarse mesh as the limit of a recursive process of subdividing each polygonal face into smaller faces that better approximate the smooth surface.

The subdivision surfaces are defined recursively. The process starts with a given polygonal mesh. A refinement scheme is then applied to this mesh. This process takes that mesh and subdivides it, creating new nodes and new faces. The positions of the new nodes in the mesh are computed based on the positions

of nearby prior nodes. In some refinement schemes, the positions of prior nodes might also be altered (possibly based on the positions of new nodes).

One of the well-known subdivision schemes is the Loop scheme, invented by Charles Loop [71]. Loop's subdivision scheme is a generalisation of C^2 quadratic triangular B-splines, and is the simplest method known to lead to tangent plane smooth surfaces. The control mesh consists of triangular faces and, like subdivision surfaces in general, can have an arbitrary topology. The resulting surface has the same topology as the control mesh.

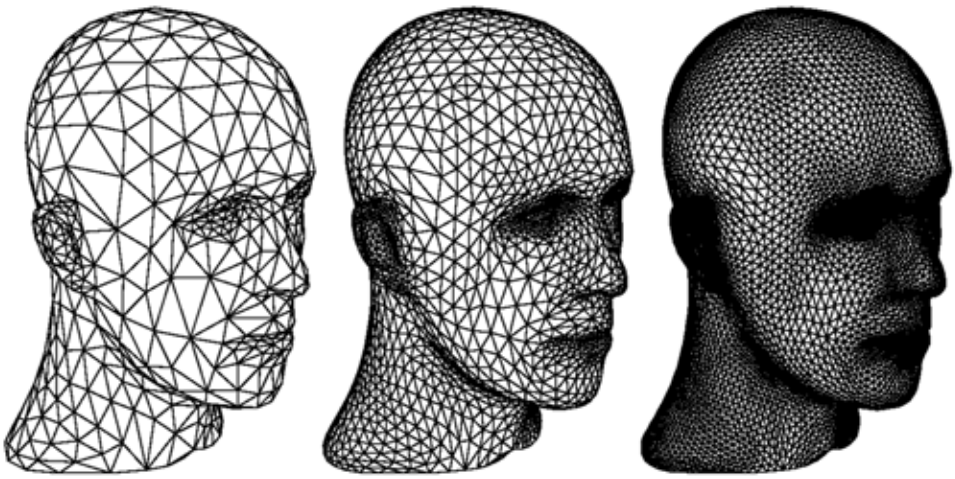


Figure 1.9: Multi-resolution subdivision surfaces [125]

One example of polygon mesh subdivision is shown in the Figure 1.9. From the left which is initial polygon mesh until right which is subdivided 4 times. This process produces a denser mesh than the original one, containing more polygonal faces. This resulting mesh can be passed through the same refinement scheme again and so on. The limit subdivision surface is the surface produced from this process being iteratively applied infinitely many times. In practical use however, this algorithm is only applied a limited number of times. The subdivision surfaces have gained popularity recently also in engineering environment and they tends to become more and more used to represent freeform shapes.

1.1.9 Synthesis on the use of geometric models along the PLC

In summary, from computer birth, the geometric modelling has been developed in different ways according to different interests, objectives and contexts. In the next section the numerical simulation by using FE meshes is presented and the motivations of this PhD thesis are further detailed.

Among the modelling methods discussed below, B-Rep and CSG as well as meshes are the most commonly used all along the product lifecycle numerical studies in industry, even if subdivision surfaces seem promising.

1.2 Basic mesh techniques

There are different categories of meshes currently used in industry in function of aim/type of study or complexity of geometric model: quadrangular and hexahedral meshes, linear/quadratic meshes, particular meshes to connect tetrahedral and hexahedral elements etc. This PhD thesis focuses on triangular and tetrahedral linear meshes to valid the proposed methods because they are simpler to generate and manipulate. If the proposed methods are validated on industrial examples, our approach can be generalised to handle other types of mesh.

This section presents the general works on meshes and notably the way they can be generated, simplified, subdivided and refined. Even if they refer to mesh modification, these operations do not really belong to the categories of mesh modifications that we targeted in this thesis.

1.2.1 Mesh genneration

Triangle and tetrahedral meshes are by far the most commonly used types of unstructured meshes. Most techniques currently used to generate such meshes can fit into three main categories: Octree, Delaunay and Advancing Front methods.

The Octree technique was primarily developed by Yerry and Mark [102, 124]. This method subdivides recursively the cubes containing the geometric model until the desired resolution. Figure 1.10 shows the equivalent two-dimensional

quad-tree decomposition of a model. This method consists in five steps: 1) Define an initial bounding box (root of the quadtree); 2) Recursively split box into 4 leaves per root to resolve geometry; 3) Find intersections of leaves with geometry boundaries; 4) Mesh each leaf using corners, side nodes and intersections with geometry; 5) Delete mesh elements outside. Irregular cells are then created where cubes intersect the surface, often requiring a significant number of surface intersection calculations. Tetrahedra are generated from both the irregular cells on the boundary and the internal regular cells.

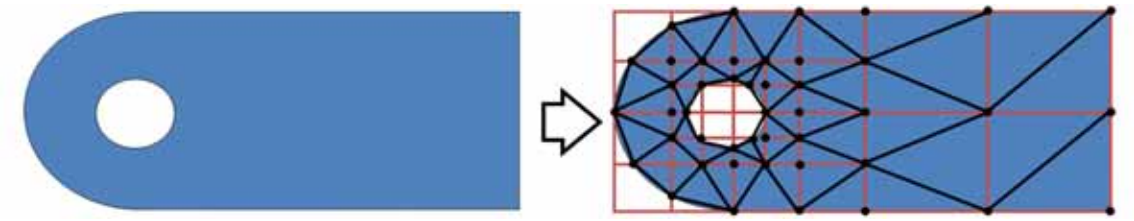


Figure 1.10: Quad-tree decomposition of a simple 2D object

Another very popular triangular and tetrahedral mesh generation algorithm is the advancing front method [67, 68, 69, 70]. In this method, the triangles/tetrahedra are built progressively inward from the boundary (edges/faces). An active front is maintained where new triangle/tetrahedra are formed. Figure 1.11 is a simple two-dimensional example of the advancing front, where triangles have been formed at the boundary. As the algorithm progresses, the front will advance to fill the remainder of the area with triangles. In three-dimension, for each triangular facet on the front, an ideal location for a new fourth node is computed. Also determined are any existing nodes on the front that may form a well-shaped tetrahedron with the facet. The algorithm selects either the new fourth node or an existing node to form the new tetrahedron based on which will form the best-shape tetrahedron.

By far the most popular of the triangular and tetrahedral meshing techniques are those using the Delaunay criterion [34]. The Delaunay criterion, sometimes called the "empty sphere" property, says that any node must not be contained within the circumsphere of any tetrahedra within the mesh. A circumsphere can be defined as the sphere passing through all four vertices of a tetrahedron. Figure 1.12 is a simple two dimensional illustration of the criterion. Since the

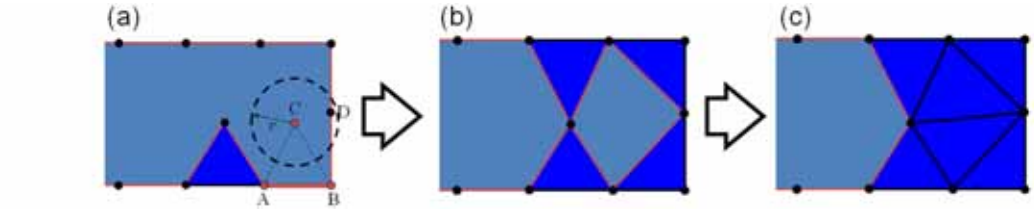


Figure 1.11: Example of advancing front on a simple 2D object

circumcircles of the triangles in left picture do not contain the other triangle's nodes, the empty circle property is maintained. Although the Delaunay criterion has been known for many years, it was not until the work of Charles Lawson [58] and Dave Watson [122] that the criterion was utilised for developing algorithms to triangulate a set of vertices. The criterion was later used meshing algorithms proposed by Timothy Baker [8] at Princeton, Nigel Weatherill [123] at Swansea, Paul-Louis George [38] at INRIA among others.

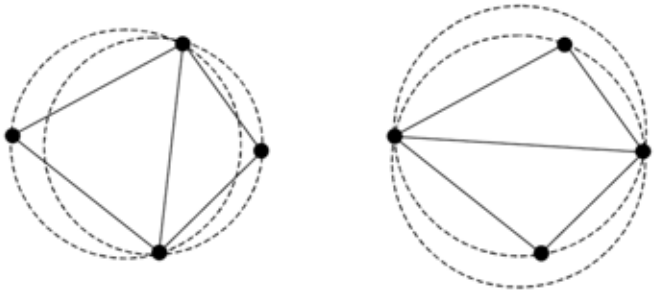


Figure 1.12: (left) Example of Delaunay criterion maintains the criterion while (right) does not

The Delaunay criterion is not an algorithm for generating a mesh. It merely provides the criteria for which to connect a set of existing points in space. As such it is necessary to provide a method for generating node locations within the geometry. A typical approach is to first mesh the boundary of the geometry to provide an initial set of nodes. The boundary nodes are then triangulated according to the Delaunay criterion. Nodes are then inserted incrementally into the existing mesh, redefining the triangles or tetrahedra locally as each new node is inserted to maintain the Delaunay criterion. It is the method that is chosen for defining where to locate the interior nodes that distinguishes one Delaunay

algorithm from another.

According to the verification of the theory and the practicality, the precision of quadrilateral mesh element is higher than triangular mesh element, so we must obtain an algorithm of how to build quadrilateral mesh. At some times in order to advance analysis precision and satisfy some special demand, we must build quadrangular mesh [21, 57].

1.2.2 Mesh simplification and refinement

In many applications, two reciprocal operations on mesh are simplification and refinement. The mesh simplification aims at reducing the geometric complexity of both polygonal and polyhedral meshes in order to decrease the load of computational systems. Reversely, there are a number of applications in which increasing the smoothness of a mesh is desirable. Surface subdivision smoothes the surface after subdividing it to improve the appearance. Examples of mesh simplification and subdivision are shown in figure 1.13. For FEA, the mesh simplification maybe useful for speeding up the analysis whereas the refinement of meshes could make the FEA result more accurate. When evaluating different design solutions for optimising a product, a first FEA could be performed with low density meshes. Then, if the results are accepted, a second FEA could be performed on the high fineness meshes FEA. This avoids losing time to do high accuracy FEA on the design solutions that are far from acceptance. Hoverer, the convergence response studies are required to validate the density of FE mesh and so, to validate the physical simulation.

1.2.2.1 Simplification

The mesh simplification, consisting in approximating a given input mesh with a less complex but geometrically faithful representation, is well-established in computer applications. Given the visual complexity required to create realistic scenes, simplification efforts can be essential to efficient rendering. Level-of-detail representations figure prominently in real-time applications such as virtual reality, topography modelling, and scientific visualisation, and as a result there is significant demand for effective algorithms for mesh simplification.

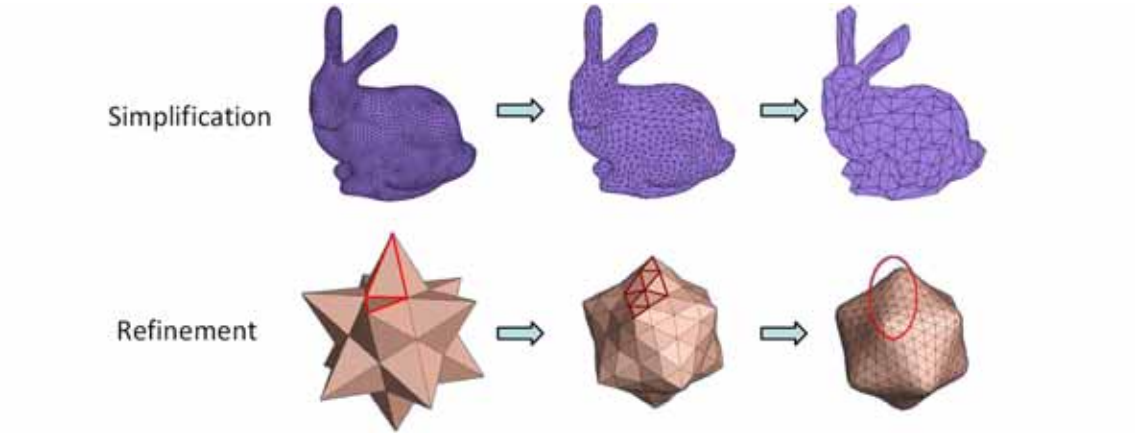


Figure 1.13: Mesh simplification and mesh refinement

Mesh simplification has been addressed in a number of publications in recent years and the strategies may be broadly grouped into two categories: 1) local strategies that iteratively simplify the mesh by the repeated application of some local operator; 2) global strategies that are applied to the input mesh as a whole. Local strategies are by far the most common. Local simplification strategies are generally greedy: vertex decimation, edge contraction, tetrahedron fusion and appearance-preserving. Global simplifications include vertex clustering and shape approximating.

Vertex decimation, first proposed by Schroeder [101], operates on a single vertex by deleting that vertex and re-tessellating the resulting hole. Typically some classification scheme based on the adjacency information of the selected vertex is used to determine the manner in which this re-tessellation proceeds.

Edge contraction, originally proposed by Hoppe [45] is the most common simplification operation. Figure 1.14 shows the proposed schema for transforming locally simplicial complex. For case of mesh simplification, the edge contraction is used. An edge contraction operates on a single edge i-j and contracts that edge to a single vertex h, updating all edges previously incident on nodes i and j to reference node h.

Many researches extend "edge collapse"-based decimation methods to volumetric meshes, focusing on accurate error evaluation metrics, while also preventing mesh inconsistencies [25, 109, 113, 114]. These methods work very well as

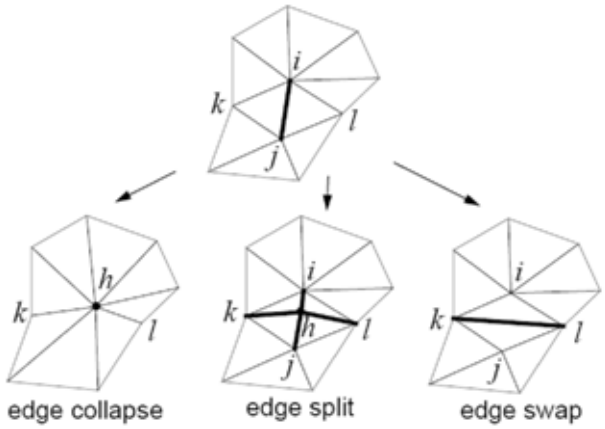


Figure 1.14: Local simplicial complex transformations [45]

metrics-guided simplification tools.

Chopra and Meyer [22] propose a fast algorithm for progressive simplification named TetFusion. The idea of this algorithm is to use a tetrahedral collapse operation in which one tetrahedron is collapsed onto its barycentre. In their work, they preserve the boundary surface by keeping all the tetrahedra on the boundary. It is faster than the edge collapse method. Figure 1.15 illustrates one example of TetFuse simplification.

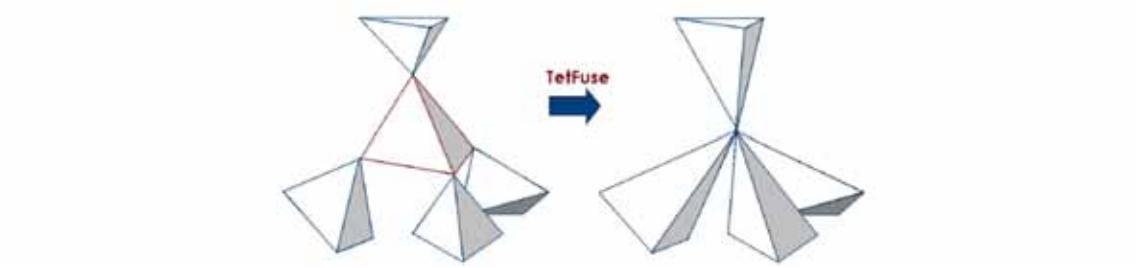


Figure 1.15: Illustration of the "TetFuse" operation [22]

Appearance-Preserving Simplification is a completely different approach to determining the error associated with a given simplification operation. It was firstly described in [28]. Their algorithm works by decoupling surface position from color and curvature information and storing the latter two quantities in texture and normal maps. A traditional geometric simplification algorithm can then be employed to filter the surface position, while a hardware-based approach

is used to filter the color and normal information, resulting in simplified representations that are nearly visually indistinguishable from the original. Papers [61, 83] described a more general image based simplification strategy that does not require specialised hardware or software algorithms. Characteristic curves of a model are detected at the variation of the shape that is computed from shading in image. The proposed simplification process will preserve these characteristics on meshes.

The method of vertex clustering was originally proposed in [16] to handle meshes of arbitrary topological structure. In their algorithm, each vertex in the input mesh is assigned a weight based on its perceptual importance: vertices adjacent to triangles with large faces and those in areas of high curvature are weighted more heavily than vertices in smooth regions adjacent to smaller triangles. Next, a bounding box is placed around the mesh and subdivided into a three dimensional grid. Finally, all the vertices in a given grid cell are clustered to the position of the vertex with maximum weight.

In [29] the mesh simplification is based on shape approximation. They employ a variational partitioning scheme to segment the input mesh into a set of non-overlapping connected regions, and then fit a locally-approximating plane to each one. The vertices on the original mesh that coincide with the intersection of three or more shape proxies are retained. An iterative edge contraction process is subsequently applied to eliminate excess geometry and produce the resulting simplified mesh.

1.2.2.2 Refinement

In order to make the FE simulation results more realistic, currently industrial companies achieve so-called convergence studies of physical model response in function of local mesh density. These studies allow choosing the optimal mesh density in order to provide accurate physical behaviour analysis, for example, local stress state assessment depending on quality of local meshing considering that geometric singularities are absent in the zone of interest. Industrial requirements in local (or adaptative) mesh refinement are then important [111].

Many numerical applications for simulating object behaviour needs a high ac-

curacy, and solid modelling in computer graphics requires partitioning geometric objects into smaller pieces. Therefore, mesh refinement algorithms have a critical role in numerical computations. Especially, 3-dimensional structures have difficulties in construction of good quality and adapted to geometry solutions.

Several approaches have been mainly used to overcome the refinement problem in 2D. The longest-edge bisection process approach guarantees a good quality, conforming mesh structure with linear time complexity [63, 90].

The approach based on the Delaunay algorithm can be summarised as adding non-vertex points in the circum-center of the worst triangles of the current structure [98]. Delaunay refinement assures the construction of most equilateral triangulation at the optimum time complexity. However, this method cannot be applied easily in 3D, and new approaches are needed for tetrahedral mesh refinement using a Delaunay triangulation based construction. Therefore, Longest-Edge Bisection method is mostly applied due to its straightforward and common implementation in the refinement process. In [60] the rough triangulation in a hole is followed by a refinement that consists in inserting barycentre nodes in triangles and swapping edges for respecting the Delaunay criterion as shown in figure 1.12.

Intersection of neighbour triangles is either a common vertex or common edge. For any triangular mesh, the longest-edge propagation path (LEPP) for a triangle is the ordered list of triangles (t_0, t_1, t_2), such that (t_i) is adjacent to triangle (t_{i-1}) by the longest-edge of (t_{i-1}) [6, 94]. Figure 1.16 illustrates the refinement of the triangle (t_0) over the initial triangulation of figure 1.16.a (with associated LEPP (t0) = t_0, t_1, t_2, t_3). The triangulations (fig.1.16.b and fig.1.16.c) illustrate the first 2 steps of the Backward-Longest-Side-Bisection procedure and their respective current LEPP (t0), while that triangulation (fig.1.16.d) is the final mesh obtained. The new vertices have been enumerated in the order they were created.

The Longest-edge Propagation Path algorithm can be generalised to 3D tetrahedral mesh. The 3D LEPP for a tetrahedron is the set of neighbouring tetrahedra that have adjacent longest-edge greater or equal to the preceding tetrahedra in the list [6, 90, 91].

The skeleton algorithm for 4-Triangle mesh refinement can be analysed in two steps: bisecting the edges in a 1-dimensional skeleton and partitioning individual

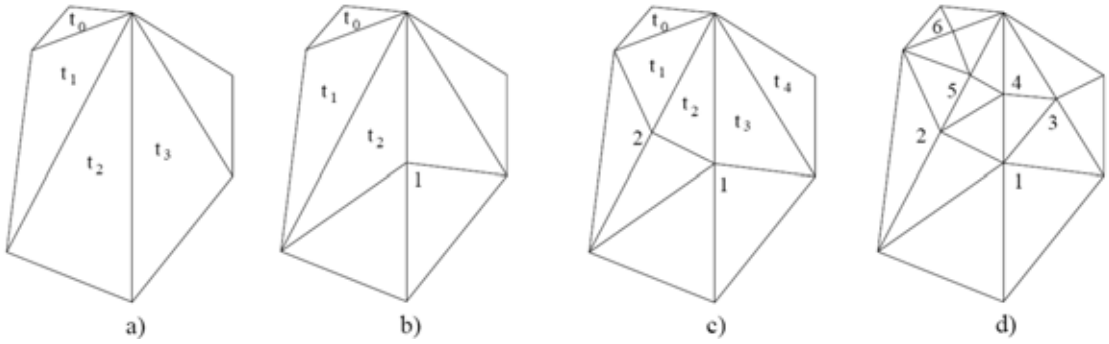


Figure 1.16: Longest-Side Bisection of triangle t0 [94]

triangles according to the bisected edges [90, 93]. The 3D skeleton algorithm is a generalised version of 4-Triangles. The 8-Tetrahedra longest edge partition is a 3D algorithm that can be explained by applying 4-Triangle skeleton refinement methodology to the faces of corresponding mesh [63, 91]. Partitioning any tetrahedron in mesh produces both conforming volume mesh and conforming surface mesh. Figure 1.17 shows one example of 8-tetrahedra subdivision schema for the initial tetrahedron (t_0, t_1, t_2, t_3). New nodes are on the middle of the edges.

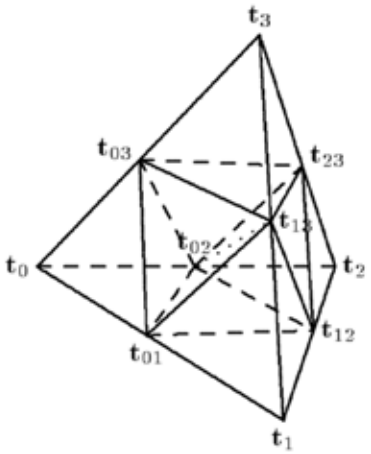


Figure 1.17: 8-subtetrahedra subdivision [91]

1.3 Numerical simulation based on FEA for product design, optimisation and maintenance studies

Nowadays, the mainstream methodology for product behaviour analysis and conceptual solution assessment relies on the following steps: conceptual phase, detailed design with CAD tools, meshing and simulation model preparing, FE simulation, result analysis and optimisation loops [1, 110].

The CAD modelling consists in creating product shape model with a certain degree of details. The meshing step converts the geometric representation of the CAD model into a so-called finite element representation thus approximating the CAD geometric representation by a discretised model. The simulation model preparing concerns the creating of specific non-manifold mesh features (cracks, duplicated mesh entities, etc.), the generation of advanced meshes composed of mixed sub-meshes (tetrahedra and hexahedra regularly jointed, adaptative/locally refined meshes, etc.), the definition of FE mesh groups (that are collections of mesh entities of different dimension), the physical description of the model such as material definition, boundary conditions, and loads definition, etc. The FE simulation requires the mesh and the physical description to compute the object behaviour evolution.

The FE simulation may predict where the product could be damaged under loads, and so prototype/evaluate a new design solution improving the behaviour of the structure. Today, the design modification of the structure is systematically performed on the CAD model of the design phase. The meshing and simulation model preparing are performed once again for performing the FE simulation on the new design solution.

However, in the context of maintenance and lifecycle problem analysis, the product is already designed, the CAD models are not necessarily available and the product behaviour has to be studied and improved during its exploitation. Companies exploiting complex installations are currently subjected to various constraints crucial from a production point of view. They can be relative to the time and cost of the production process stops, to the efficiency of maintenance

solutions, to production safety criteria, etc.

For example, in the field of power production, it is critical to identify the problem source and to provide the appropriate solution while taking care of the triptych: Time, Quality and Cost. As a reference, for the Electricit de France (EDF) Group, the total cost of one day of stop of a nuclear power station represents several hundreds of thousands of euros.

The expertise of problems taking place on EDF production sites corresponds to the analysis of normal functioning of production equipments. Such expertises generally concern planned preventative and/or occasional maintenance of machineries, lifecycle assessments or some improvements of the production equipment behaviours (e.g. mechanical, acoustic or hydraulic behaviours). They have to satisfy various constraints like:

- to guarantee the continuity of production and to avoid the stop of production cells,
- to improve the equipment functioning, to propose the optimal exploitation regime,
- to answer the production safety criteria (e.g. in the field of power production), etc.

Taking into account all these constraints is particularly important during operational studies where it is critical to provide quickly the optimised solution and to ensure its effectiveness. So, it is necessary to reduce the time/cost of numerical study, i.e. to reduce the time of different study stages.

In the case of behaviour improvement studies applied to the maintenance/lifecycle context, some local structural modifications are generally possible. For example, in the field of power production, it is not allowed to propose global modifications on the production installations subjected to specific exploitation constraints and provided by the specialised equipment certified designer. Moreover, each structural change, even a small local modification, requires product cycle stops and validation expertise accomplished by competent safety authorities. Hence, the solutions requiring significant modifications requiring long stops of production sites and/or complete review of a new production cycle are not realised.

1.3.1 Classical CAD-FEA loop

The preparation of digital prototypes for FE analysis is generally carried out by applying the following four-step loop: development of the CAD mock-up, generation of the meshes, preparation of the FE mesh models suitable for the adapted for a given FE analysis code and the FE simulation. Classically, the evaluation of alternate solutions improving the product behaviour requires several updates of the initial CAD model and the consequent repetition of the above processing steps on the new CAD version. According to the traditional prototyping method, this evaluation would include in outline the 4 following stages:

1. CAD model design
    - development of the complex CAD model which does not exist, it could be newly designed or modified from an old CAD model.
    - modification / adaptation of the CAD model for making it meshable.
2. MESH model creation
    - development of an advanced mesh model taking into account different aspects such as mesh quality criteria, variable densities of the mesh etc.
3. FE SEMANTIC insertion
    - creation of mesh entity groups on which different FE simulation semantics will be defined in the following step. There exists two ways to create them: either manually while selecting a set of mesh entities or more automatically while selection a geometric partition of the CAD model and then transferring this information during the mesh creation. This step requires a very good skill and is time-consuming.
    - Assignment of FE simulation semantics corresponding to the mesh groups that are defined in the previous step:
        - describing mechanical links between the 1D and 3D model partitions geometrically separated,

  - characterising several materials,
  - describing specific geometric/mechanic parameters (beam modelling, spring discrete element modelling, punctual mass), defining BCs and different loads.

4. FE SIMULATION

   - FE simulation model definition using a given FE code and its execution to calculate the behaviour of the structure;
   - improving of FE simulation model by testing of various methods to describe more accurately the physical behaviour of the structure;
   - tuning/validation of the FE model through experimental results. It consists in doing an experimental test and in comparing the results obtained from numerical simulation and real measuring. The goal of such a tuning is to find accurate values of unknown parameters sensible from computation point of view in order to obtain a FE model response close to experimental data. If the difference is important it's necessary to return to some stages of the process to valid the numerical model.
   - proposal of new design solution for improving the structure behaviour if the old one is not acceptable;

At end of the stage (4) we can prototype possible solutions and so, to realise envisaged model modifications. Hence, several new loops from the stage (1) to (4) should be realised in order to evaluate the corresponding solution.

1.3.2 Industrial case studies using classical CAD-FEA loop

To illustrate clearly these steps let us consider a industrial case study in figure 1.18 that presents an example of complex models the EDF engineers have to deal with. Figure 1.18 shows the model of a quarter of caisson in which a structural modification has to be performed. It corresponds to a local caisson model developed for fast structural behaviour analysis with prototyping of an envisaged solution.

The figures 1.18.a - 1.18.d shows one loop of numerical study for the initial design model and three modifications must be analysed (fig.1.18.e - fig.1.18.g). The assessment of each modification is based on a loop similar to the one for the initial model study. The number in a circle indicates the loop number in each simulation case.

Loop (1) : at the first loop the caisson model is designed.

1. CAD stage: Creation of the CAD model of the caisson (fig.1.18.a).

2. MESH stage: Meshing of the CAD model. The mesh of caisson shown in the figure 1.18.b is a complex FE model, containing 1D/3D elements and adapted for a particular FE analysis; different mesh densities are specified for different areas according to specific simulation interest.

3. FE Semantic stage: Creation of groups containing mesh entities associating additional information required for FE simulation. For the case of caisson shown in figure 1.18, there are 30 mesh entity groups created which are distinguished by different colors in the Figure 1.18.c. Various mechanical semantics required for FEA are then defined and associated with the corresponding groups.

4. Simulation stage: FE simulation to analyse the structure behaviour. A local stress situation as well as a possible cracking of the damaged stiffeners are studied in the present example (fig.1.18.d).

Loop (2) : According to the simulation result of the first loop the first modification consisting in introducing the crack feature to simulate its propagation is prototyped during the second loop of the study. Therefore the four stages of simulation loop similar to the first loop are repeated: CAD preparing by modifying the previous CAD model (fig.1.18.e), meshing, FEA semantics insertion and simulation. In this loop, the modification proposed on the CAD model consists in making a planar fissure according to the crack plane (fig.1.18.e).

Loop (3) : According to the simulation result of the second loop showing that the potential crack in the stiffener may damage the caisson and so, stop the production process, a proposal of structural modification corresponding to a

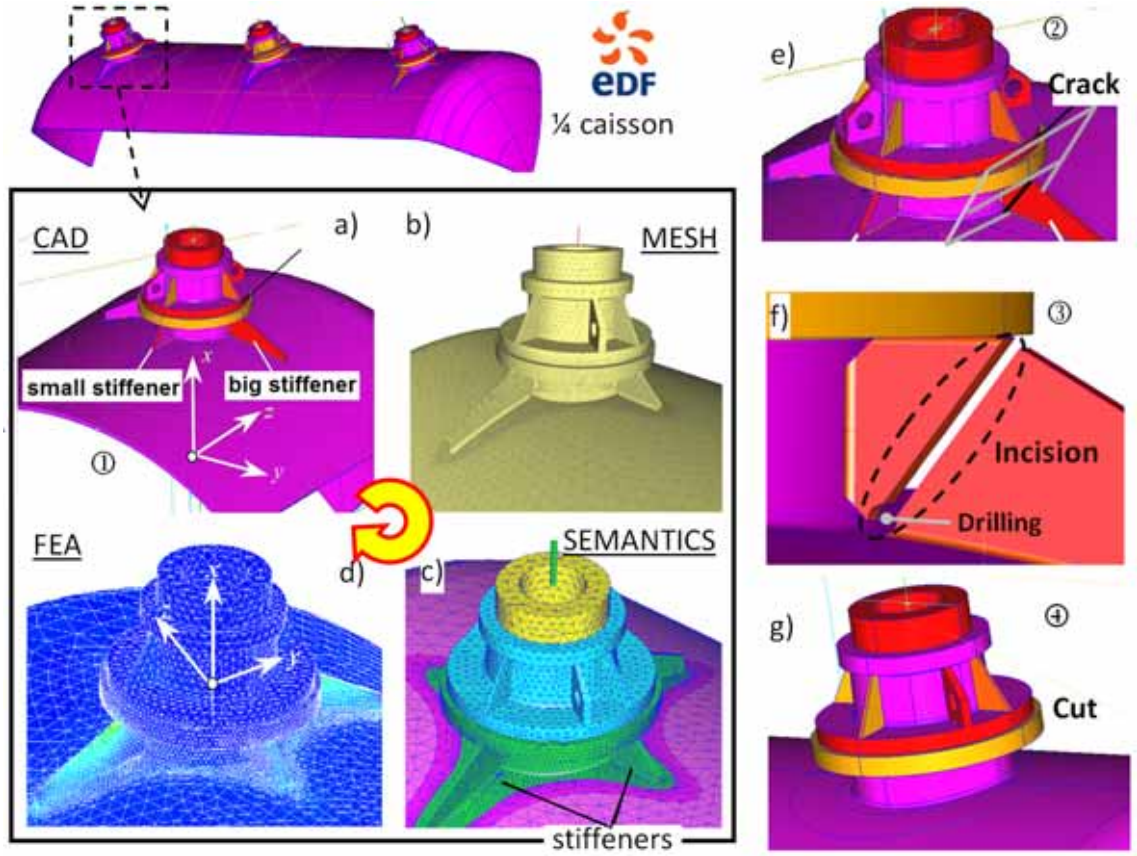


Figure 1.18: Example of 4 steps simulation loop (a,b,c,d) to introduce and analyse possible local structural modifications of a caisson (e,f,g) (courtesy EDF R&D)

possible design solution is prototyped during the third loop of the study. The CAD model is created by incising and drilling the stiffener on the initial CAD model created during the first loop (fig.1.18.f). For evaluating this modification, all the other three stages similar to the other loops are repeated to complete the simulation.

Loop (4) : According to the simulation results of the third loop, it is still necessary to propose another structural modification that is prototyped during the fourth loop of the study. In this last loop, the stiffener of the initial CAD model created during the first loop is completely cut away (fig.1.18.g) in order to avoid the local stress phenomenon. This modification solution is adopted at the end of four loops where all the stages introduced in Loop (1) have been repeated

several times.

In this example of maintenance study, three structural modifications are prototyped and two design solutions are proposed and evaluated by using the FEA loop in four steps. These local structural modifications are proposed in an order of numerical study progress. Several CAD models and FE models are created and consequently the related FEA semantics are defined several times for performing the simulation.

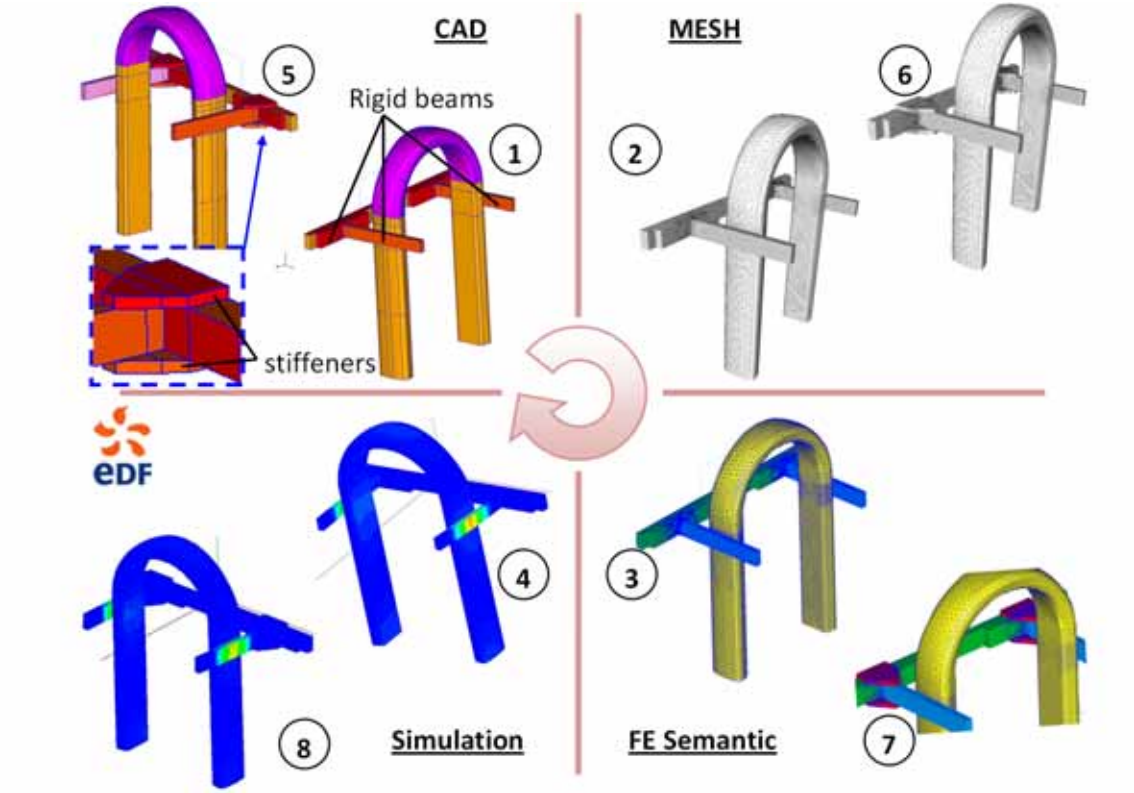


Figure 1.19: Numerical assessment of a new solution based on the classical product optimisation method: stiffener addition to a U-like testing bench model (courtesy EDF R&D)

~~In this example of maintenance, three structural modifications are proposed and evaluated by using the FEA loop in four steps. These local structural modifications are proposed in an order of increasing difficulty of their application on the real model. The modification of the easiest applicable on the real model is preferable as long as the corresponding FEA result is acceptable. Several CAD~~

This paragraph is a redundant.

models and FE models are created and consequently the related FEA semantics are defined several times for performing the simulation.

Figure 1.19 illustrates another industrial example of fast FEA study prototyped by the EDF engineers in which a structural modification (new design solution) consisting in new part addition is accomplished by using the classical numerical simulation method. It concerns the analysis of the behaviour of a U like testing bench on which there are several beams. During the first loop, the CAD model is designed (stage 1) and meshed (stage 2), semantic data (groups and associated information) are defined (stage 3), and the FE simulation model is created and the simulation is performed with the corresponding post-processing analysis (stage 4). According to the simulation results an envisaged modification consists in adding stiffeners in the zone of weld contacts between the beams as shown in figure 1.19 (a zoomed view is made in the blue dashed rectangle, two stiffeners are added on and below the contact zone). This corresponds to a new loop of the numerical study and CAD modification (stage 5). Then, the whole model is re-meshed as shown in figure 1.19 (stage 6), all groups are re-created for defining FE semantic information necessary for FE simulation (stage 7), and a new FE simulation and post-processing of results are performed (stage 8). Here, the type of structural modification is different from manipulations in the case of caisson (fig.1.18). The example of caisson concerns a material removal operation reversely the operation applied to the U like testing bench model is a material addition operation.

1.3.3 Notion of groups and type of FEA semantics

It seems quite clear from the presented examples of operational industrial studies that the multiples returns to the CAD model are not appropriate to quickly implement and assess a local structural modification. This is especially true when the model contains a huge amount of semantic data involving several mesh groups. For example, some EDF models can contain up to 500 mesh groups necessary for FE analysis (BCs, link relations, different behaviour laws, geometric parameters, mechanical modelling of specific phenomena, etc.) as well as particular post-processing. More complete description of the BCs is given in chapter 5 (p.147)

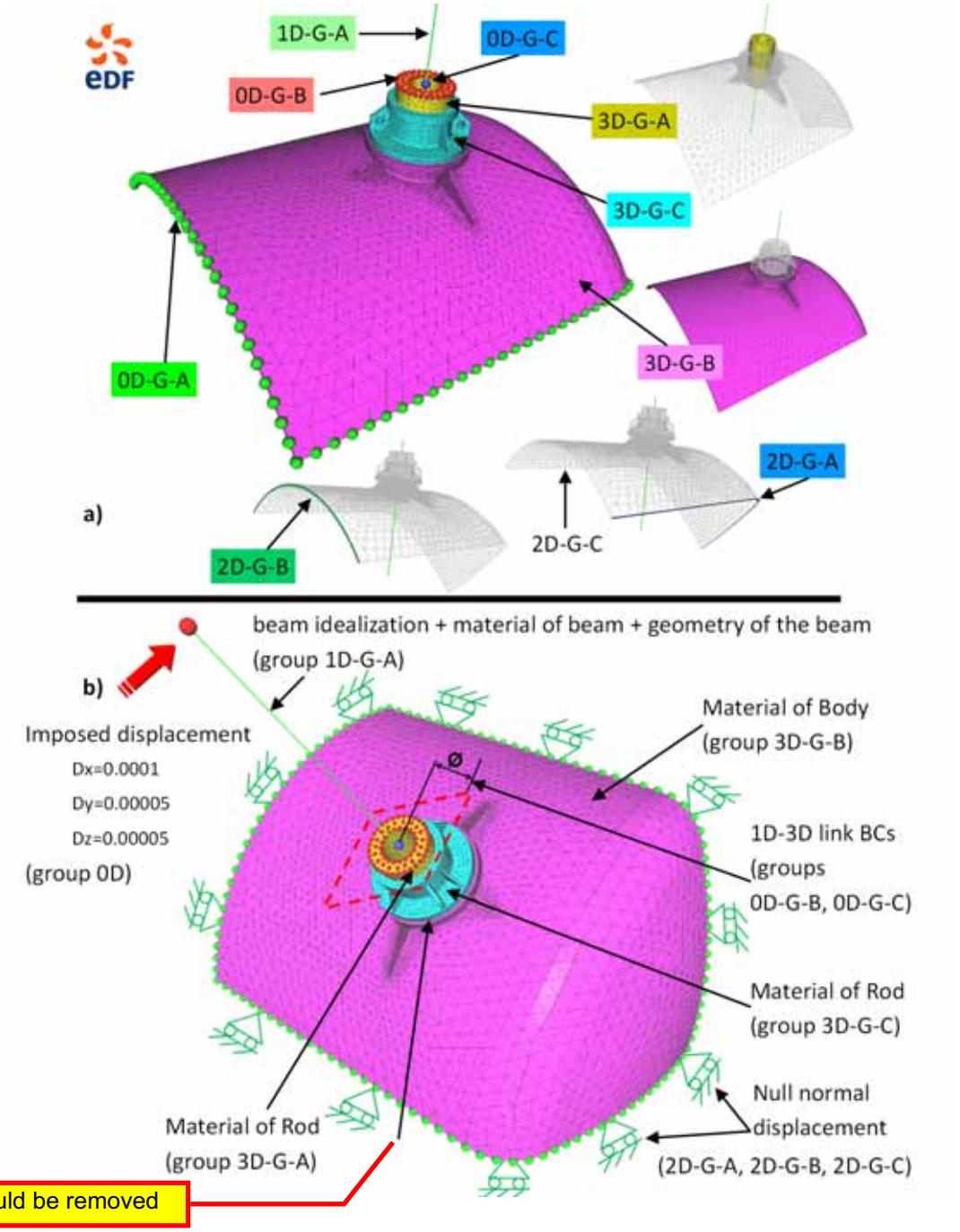


Figure 1.20: Groups and associated FEA semantics defined on the CAISSON (courtesy EDF R&D)

together with other semantics relative to the shape of the elements, the material that is used and so on.

Figure 1.20 shows different mesh groups defined on the Caisson model (fig.1.20.a) and the related associated FEA semantics (fig.1.20.b). The dimensions of the groups are also various: 0D (group of nodes), 1D (group of edges), 2D (group of triangles) and 3D (group of tetrahedra). Some groups are created automatically by using geometric partitions in the CAD model. The meshed model of the caisson in figure 1.20.a has different "partitions" corresponding to 3D element groups, which are distinguished by different colors. Some other mesh groups are created by selecting manually sets of mesh entities or by using "geometric" groups defined on CAD level.

- From volume (3D) point of view the model is subdivided into two tetrahedral groups "3D-G-A", "3D-G-B" and "3D-G-C" (fig.1.20.a). These groups are used to define different materials (fig.1.20.b).
- In surface (2D) aspect two groups of triangles "2D-G-A" and "2D-G-B" are defined on the lateral boundary of the volume group "3D-G-B" (fig.1.20.a). These surface groups in a 3D mesh are used for modelling BCs, in particular, conditions of symmetry in displacements to fix the given structure.
- Linear (1D) groups are also involved by considering edges. The Group "1D-G-A" corresponds to a set of edges (fig.1.20.a) and used to idealise a beam like component (fig.1.20.b).
- There are also node (0D) groups. For example one node at extremity of the isolated edges is belonging to the group "0D-G-C" (fig.1.20.a). The group of nodes "0D-G-B" is located on the top of the cylindrical part (fig.1.20.a). BCs semantics of staying on the same plane is defined between the node in the "0D-G-C" and the nodes in the "0D-G-B" (fig.1.20.b). Another node group is defined for the node on the top extremity of the isolated edges and one semantics, imposed displacement (i.e. displacement BC) (D_x, D_y, D_z), is defined on it (fig.1.20.b).

The FEA mesh of the model U-like testing bench shown in the figure 1.19 has also numerous groups and FEA semantics defined (fig.1.21):

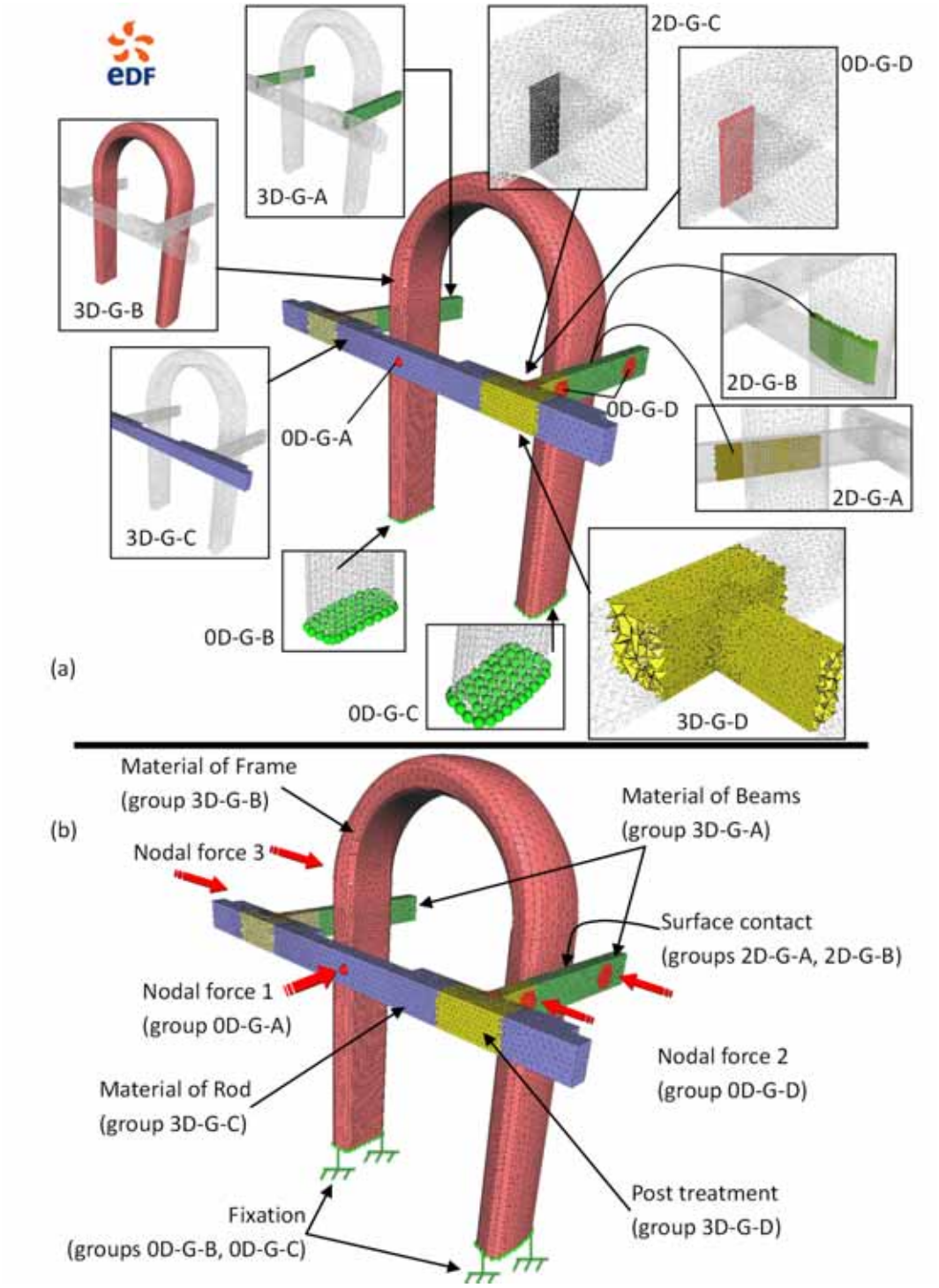


Figure 1.21: Groups and associated FEA semantics defined on the U-like testing bench (courtesy EDF R&D)

- Concerning the 3D groups the mesh is subdivided into three tetrahedral groups “3D-G-A”, “3D-G-B” and “3D-G-C” (fig.1.21.a). Different materials of the different parts (beam, rods and frame) are then associated with these groups (fig.1.21.b). The tetrahedral group “3D-G-D” is used for local post-processing the FEA results solely in the area covered by the group.

- A 2D group made of triangles, “2D-G-C”, is present between the two volume groups with semantics “Beam” and “Rod”. This group contains the triangles on the contact interface of the two parts. The two triangle groups “2D-G-B” and “2D-G-A” (fig.1.21.a) are respectively on the volume parts “Frame” and “Beam” (fig.1.21.b). The two triangle groups are used for supporting the FEA semantics of type “contact BCs” between the two volume parts.

- In this model no edge groups are present.

- The groups “0D-G-B” and “0D-G-C” containing all the nodes at the bottom of the U-like frame part (fig.1.21.a) are used to define BC semantics of fixation of the model (fig.1.21.b). A FEA semantics “Nodal Force 1” (fig.1.21.a) is defined on the unique node of the group “0D-G-A”. This group allows modelling vibration force when the testing bench is loaded. Two other symmetric nodal forces “Nodal Force 2” and “Nodal Force 3” are defined separately on the two beams. The “Nodal Force 2” is associating with the node group “0D-G-D”. These nodal forces simulate clamping forces due to bolts allowing creating contact forces between “Frame” and “Beam” (bolts are not geometrically modelled).

The presented groups are used to model mechanical behaviour of the testing bench before any modification. Added stiffeners in the zone of weld contact between beams and rod will modify locally the FEA semantics: new FE mesh groups are created end corresponding physical semantic information is defined.

1.3.4 Model modification operation categories

According to the different industrial case studies using FEA, three main classes of geometric model modifications corresponding to current industrial needs have

been identified from a topological point of view:

- material deletion operation,
- material addition operation,
- material cut operation (for exemple, cracking / contact zone feature insertion).

The material deletion operation mainly concerns the modifications which remove material (e.g. volume in case of 3D model) from the model, like cutting, incision, drilling, filleting, chamfering and so on. In the EDF study example shown in figure 1.18 the operations of incision, drilling and cutting are performed. At the end, these operations produce a loss of material.

The material addition operation concerns the modifications which add material (e.g. volume in case of 3D model) to a reference model, like merging, filleting, chamfering and so on. In the EDF study example shown in figure 1.19 the merging of two 3D models is performed.

The filleting operation can delete material as well as add material. Figure 1.22 illustrates a filleting performed on two sharp corners. Figure 1.22.a presents the initial model, while figure 1.22.b shows the part after the two filleting operations: the filleting of the inner corner gives rise to an addition of material, while the filleting on the outer corner returns in a material removal.

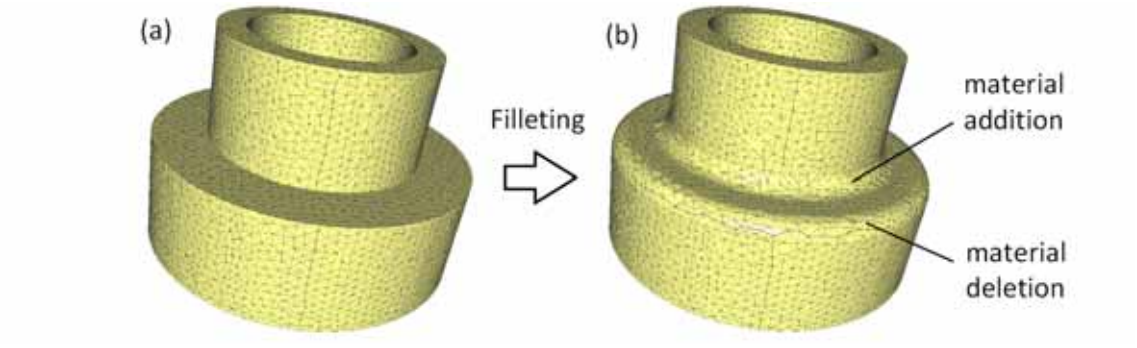


Figure 1.22: Example of inner / outer corner filleting

The material cracking or cutting, also refered as contact zone feature insertion, is another local structural modification that we do usually on the FE model. The

difference between material deletion and material cutting is that during the first operation a part of the material is removed while during the second operation, the material is cut but not removed and all cut parts are kept in the model. Figure 1.23 shows an example in which a stiffener is cracked. The crack feature zone is modeled using double entities (nodes, edges and faces are duplicated). Such a feature allows simulating contact problems and crack phenomena (assessment of damage risk, crack propagation). It can be mentioned that several FE groups are generally created in order to apply contact BCs as well as to facilitate the post-processing. Figure 1.23.b shows the result of the FE simulation wherein the crack is opened. Today, to introduce a contact zone into a mesh, it is necessary to go back to the CAD model on which an internal edge/surface must be inserted. Then, the modified CAD model must be totally re-meshed and all previously defined mesh FE groups must be re-created.

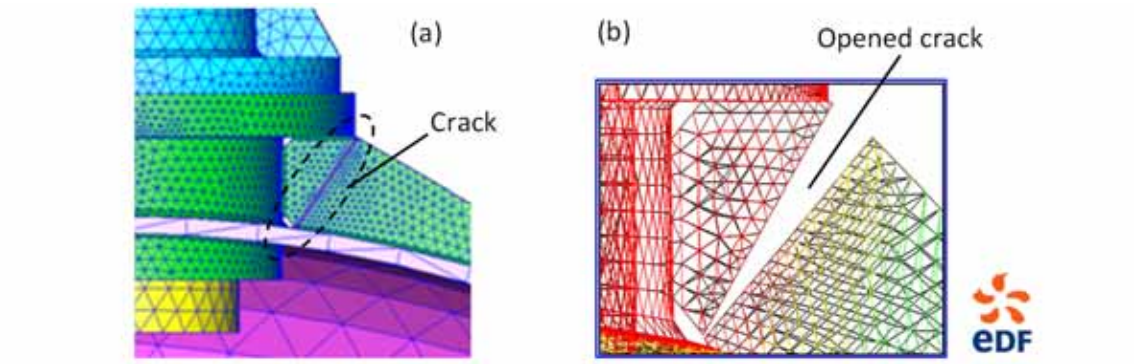


Figure 1.23: Crack insertion into the mesh model in order to model a crack phenomenon (courtesy EDF R&D)

1.3.5 Specificities in some industrial simulation contexts (maintenance, Reverse Engineering ···)

The classical FEA loop in four steps is widespread in industrial design companies. However, its use is not always obvious during the fast studies applied to maintenance/lifecycle problems. Indeed, it requires the access to the design history, also called the construction tree, of all the components forming the CAD model, and this to be able to modify it. This is not always possible, particularly when

the geometries are tranfered using neutral formats (e.g. IGES or STEP files) or when the CAD model is reloaded with different versions or releases of a given CAD system.

1.3.5.1 Lake of time in the re-creation of specific meshes

When undertaking a modification, it is important to guarantee the meshing capability of the complex CAD model while taking into account different quality criteria of the FE mesh as:

- size/shape of the elements generated, good approximation of model details,
- free/mapped and/or adaptive mesh partitions linked (conformity of the mesh model when non-regularised partitioned CAD models are used),
- absence of local geometric singularities critical for FE simulation (computing of the stiffness matrix, realistic evaluation of a local stress state).

Furthermore, some complex meshes of large size adapted to the advanced FE simulation might contain many different entities difficult to model within the CAD mock-up:

- double mesh entities (e.g. 2D elements, nodes) to model contact and multi-physics interaction problems (e.g. soil-structure, fluid-structure, etc.),
- mesh entity groups to define mechanical/geometric parameters as well as to apply specific mechanical relations or BCs required for a given FE analysis,
- specific groups to model a complex phenomenon (e.g. crack, contact problem).

Therefore, if the designer has to go back to the CAD model, he/she loses this specific information, mandatory for advanced FE simulation, and that will have to be redefined later on.

1.3.6 Absence of the CAD model (starting from scratch)

One critical situation is that when analysing real configurations, the CAD models are not always available or not uptodate with the real component to be studied. Actually, very often when doing maintenance operation on real installations that have been bought to another compagny, you don't have the corresponding CAD models. Therefore, it is possible to reverse engineer the product which does not necessarily require the complete recreation of a CAD model. In this case, the simulation is directly performed using the discrete representation.

Digitalisation of the industrial structure is not a panacea and represents only an alternative solution to create the appropriate geometric model required for a given FE simulation. The cost of digitalisation process as well as the quality of obtained results represent main problems for fast industrial studies. Therefore, the re-creation of CAD model adapted to a given FE simulation is usually privileged by companies. In this case, the CAD model can differ from design model elaborated by the equipment manufacturer: the CAD model adapted to the maintenance context corresponds to a local (so-called "zoom") model representing the zone of interest only, contains idealised components, represents all main parts interesting from FEA point of view (i.e. geometry is simplified). Furthermore, the improvement of CAD model specifically adapted to the FE simulation is required if different internal components having various material properties have to be represented; the external skin of the digitalised object does not allow to access to the interior of the 3D structure. The meshed model based on the created simplified CAD adapted for FE simulation has to be tuned in order to represent accurately the physical behaviour of the structure. It can be mentioned that the re-created CAD model allows simplify the first loop of optimisation/design study.

1.3.7 Tuned meshes

CAD models mainly consider the object to be studied as perfect that does not necessarily correspond to the reality. For example the plane on the CAD model is mathematically planar whereas the corresponding face on the real model is not necessarily planar since it can be deformed due to the long time using. As a consequence, in order to improve the mechanical behaviour modelling of real

structures, the mesh models are often tuned to better fit what can be measured "on-site" if such measurements are possible.

The tuning process is performed using experimental data. The deformations of the real model under a certain load are measured. The deformations of the virtual model under this numerally modelled load are also computed. If the differences between the real and numerical results are under a certain tolerance, the simulation model is certified to represent the mechanical behaviour, otherwise the model is not accepted and it should be modified or tuned until it better fits the real behaviour. The tuning of a model refers to both geometric and physical adjustments, acting for example on the shape of the model, the density of the finite elements, the material behaviour law, the BCs, the mass and stiffness of idealised/simplified components and so on.

The model tuning is a very long and expensive process. Design modifications may be prototyped to improve the physical behaviour of the structure can affect or not the mechanical parameters tuned during the FE simulation. For local design modifications, it can be assumed that such modifications are not affecting significantly the tuning of the simulation. This should be possible since most of the time the considered modifications are local and do not affect the entire model.

Therefore new tools to operate quickly so-called "dead" models (CAD, meshes) are required.

1.3.8 What would be a possible solution generally speaking ?

Current commercial CAD systems do not make it possible to automate the process of direct complex mesh modification while preserving semantic data added at different stages during successive model transformation and enrichment phases. As a consequence, during the prototyping of structural modifications, even small local changes require expensive complete re-creation of the FE mesh model that is critical for fast studies. In order to overcome these limits, we propose a fast prototyping framework working directly at the level of FE mesh enriched by simulation semantics. This corresponds to a CAD-less shape modification approach allowing the engineer to directly operate on the meshes containing an arbitrary number of

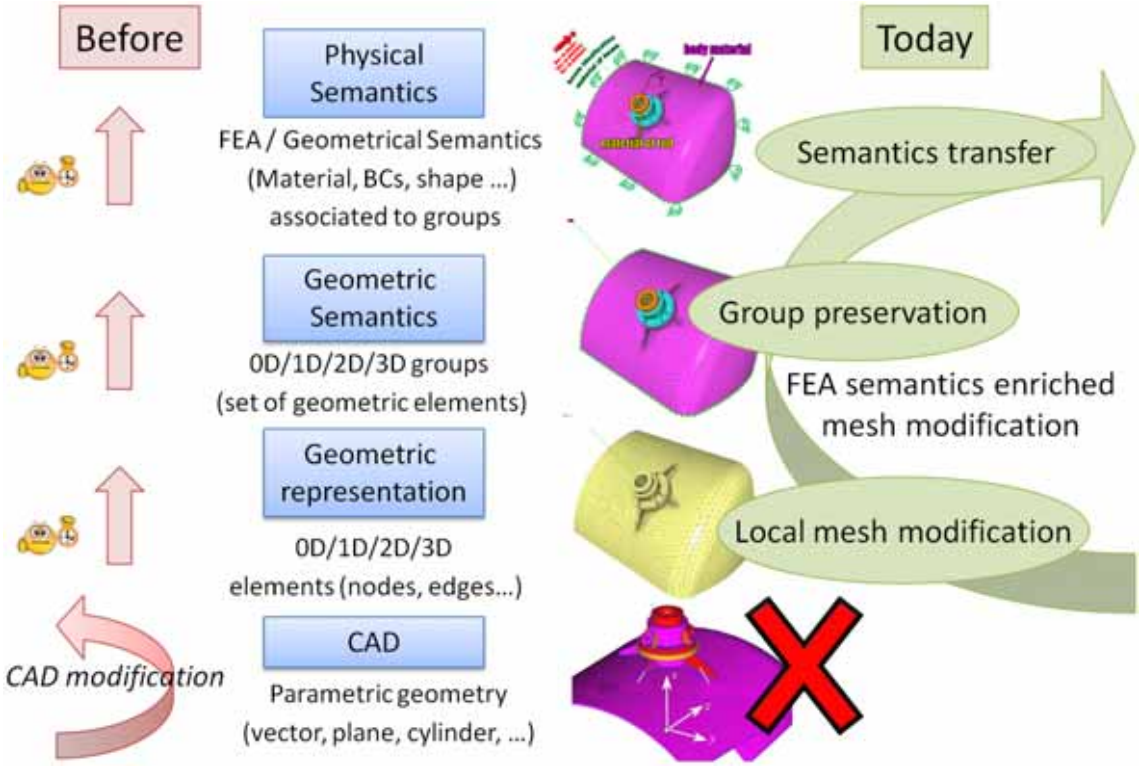


Figure 1.24: Workflow for FE mesh model preparation (courtesy EDF R&D)

semantic data already inserted and validated. Figure 1.24 shows clearly the comparison between the current classical workflow for FEA model preparation (on the left side) and the proposed workflow (on the right side). As described in section 1.3.1, today many efforts are required to apply model modifications on the CAD model and it takes time for re-creating the mesh model, re-creating the different mesh entity groups and re-associating mechanical semantics. Whereas, with our proposal, engineers can directly modify locally the FE mesh model while preserving existing mesh groups and transferring different semantics required for FEA. For achieving this objective, different aspects should be taken into account: mesh topology modification and deformation, group handling, semantics maintenance and transfer. Different aspects related to meshing and direct mesh modification are discussed in chapter 2.

One industrial example to illustrate the necessity to take into account the existing semantic information to accelerate the new design solution assessment is

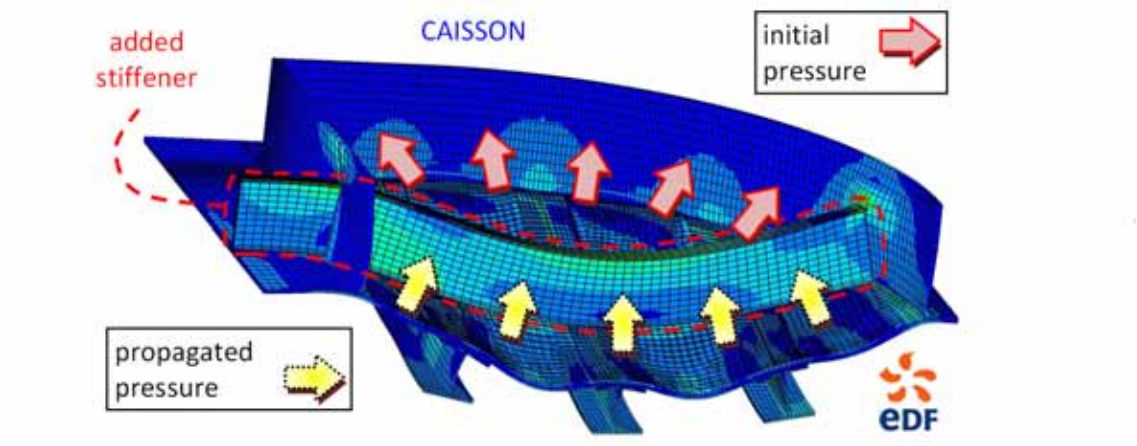


Figure 1.25: Example of stiffener addition onto a Caisson model (courtesy EDF R&D)

given on figure 1.24: we want to add a stiffener on the mesh of a caisson (fig.1.25) to improve quickly the product behaviour with respect to already defined BCs (fixing and internal pressure in the present case). Our solution is to merge the mesh of a stiffener, modeled separately, to the existing mesh of the caisson as well as to keep all semantic information useful for FE simulation. The merging should work at two levels: the geometric and semantic levels. First, the merging process should modify locally the contact zone of the two meshes while maintaining the groups associated to the various elements. Then, the merging process should suggest propagating the fluid pressure that is defined initially on the caisson interior wall, onto the stiffener external surface. Of course, this process should be semi-automatic in the sense that the engineers should always keep the control on the semantic treatments. As consequence, there will be no need to re-select the group entities and it will be possible to perform directly the simulation on the new structure allowing to quickly understand whether the modification is good or not.

1.4 Conclusion

In this first chapter, the general context of this PhD thesis is presented. Firstly, different geometric models used during the product lifecycle are presented. Sec-

ondly, the industrial context is also introduced and illustrated with several examples. In industry, the product design and re-design are both applied with a classical FEA workflow. With this workflow each time the product is re-designed under a CAD system the entire FEA workflow stages are repeated in order to evaluate the product design solution. Therefore, for evaluating different design solutions even having slight difference the time-consuming full FEA workflow should be repeated. In addition, this classical FEA workflow prevents different product optimisation solutions from being evaluated through the acquired mesh model of an existing physical model because of the absence of its CAD model. It has been demonstrated that the classical design approach based on CAD returns is not optimal in the context of maintenance/lifecycle analysis when the fast assessment of solution improving the behaviour of the product is crucial in order to avoid expensive production stops.

Therefore, the needs for developing new CAD-less modification operators are straightforward. Not only the geometric model has to be modified, but also the groups gathering together mesh entities and supporting FE simulation semantics have to be updated accordingly.

Chapter 2

Modification of enriched FE meshes

In the previous chapter we illustrated some industrial needs in modification of FE mesh models. In this chapter, a state-of-the-art on mesh modification methods is proposed. Several criteria impacting the modification of FE mesh model are then introduced in the context of product maintenance study. Two types of criteria are used to ensure the quality of modified mesh used in FE simulation: geometric and semantic criteria. Various approaches are then introduced according to several issues in mesh modification taking into account various mesh quality criteria: meshes intersection, crack feature insertion, cutting, sharp edge filleting, knowing that the limits between those categories are not always well-defined by considering the multitude of industrial needs, sometimes very specific. The manipulations of semantics are categorised into annotation and transformation.

2.1 Criteria for FE mesh modification methods

The modification methods of mesh models have been largely studied in literature. The geometric models concern the ones used in different domains from computer visualisation to physical FEA. The modifications performed on visualisation type models are not acceptable for FEA. This is due to the fact that the mesh model used for FEA should be paid attention to the quality of mesh elements such as the

aspect ratio that is very important to ensure a good accuracy of the simulation results. In addition, the initial model shape should be as close as possible to the real shape of the product so that the results are as close as possible to the real behaviour. All these geometric quality aspects to be respected when modifying a FEA mesh.

In the context of the fast prototyping of alternate solutions during the product optimisation, the geometric model has already been used once for the FEA before its optimisation. Therefore, there might be much more attention to be paid when modifying geometric model since several adjustments have certainly been performed on it and some mechanical semantic information have certainly been attached on it. In order to keep the validity of the geometric characteristics (e.g. shape, size adjustment, non-manifold elements), the mesh modification performed should be as local as possible and the modification area should have its geometry characteristics as close as possible to the original one. In the unmodified area, it is therefore possible to keep the links between the mesh elements and the concerned semantic information. In the modified area, the semantic information has to be preserved and updated so that each prototyped solution does not necessitate the re-definition of semantics.

2.1.1 Geometric criteria

The geometric criteria that should be respected during the mesh modification are:

- **Local modification** (a): the modification should influence as few as possible the model so that the geometrically and/or mechanically validated FE mesh model is perturbed as little as possible.
- **Initial shape of the model** (b): a mesh model could be deformed, cut or added material, the shape (external skin) of the mesh model should still correspond to the initial shape in the unmodified areas.
- **Quality of the mesh elements** (c): this criterion is used to qualify the modified mesh elements in terms of aspect ratio.

- **Self-intersecting elements** (d): this criterion is used to qualify the capability to avoid self-intersections when performing mesh modifications.
- **Shape of the modification tool** (e): this criterion is used to qualify whether the shape of the modification tool is satisfied or not when performing mesh modifications. For example when drilling a mesh model with a cylindrical surface that is considered as the modification tool, after operation there should be a part of the external skin on the mesh model approximating to the cylindrical tool.

2.1.2 Semantic criteria

The semantic criteria that should be considered during the mesh modification are:

- **Definition of groups** (f): this criterion checks whether the group definition on the modified mesh elements are maintained or not. It concerns the shape of the group represented by the associated mesh elements and the link with concerned mesh elements. Actually the group definition should be completely preserved for the area without mesh modification and the group definition should be updated in the mesh modification zone so that the group definition is as close as possible to the initial group definition.
- **Definition of semantics** (g): this criterion checks whether the semantic information associated with the different groups are preserved during the mesh modification or not. This criterion also evaluates whether an update of the groups involves an update of the semantics, which may happen in some configurations. The updating of physical semantics can be automatic or semi-automatic, and depends on the nature of semantic information. The expert has to assist and valid the semantic information propagated after mesh modification.

2.1.3 categorisation of mesh modification methods to be studied

Mesh modification has been subject to various researches since many years and it has become even more important recently due the improved computation capabilities and the diffusion of specific applications, such as animation movies, gaming, handling of scanned objects and Virtual Reality. Recently several modelers working directly on meshes came out. They generally offer functionalities to create shapes through the instantiation of simple primitives and successive deformations (SALOME, I-DEAS, NX, FEMAP, etc.). Because of the context in which they are used, they generally do neither care about the quality of the obtained mesh elements nor about the preservation of the possible associated semantics.

Different existing methods of mesh modification are studied and analysed regarding to the different criteria raised previously. According to different needs in terms of mesh modification in the context of industrial product maintenance, the state-of-the-art has been categorised as follows:

- Meshes intersection
- Crack insertion
- Cutting operation
- Mesh filleting
- Semantics manipulations

In the following sections, these approaches will be analysed according to the criteria introduced in the previous section (from a to g). If some works represent advantages with respect to a certain criteria, the symbol $^{\oplus}$ will be used after the concerned criterion, otherwise if the proposed approach is not good with respect to this criterion, the symbol $^{\ominus}$ will be employed. Such a comparative analysis takes into account different requirements from FE application point of view.

2.2 Mesh intersection

Boolean operations are performed by giving two operand meshes. The Boolean operations are union, intersection and difference. In [55], Boolean operations are performed on volume mesh by doing intersection of boundary meshes and completely re-filling the tetrahedral mesh. The full re-meshing allows producing good quality mesh elements ($c^{\oplus}$) but this is not admissible when manipulating tuned FE meshes that can be only modified locally ($a^{\ominus}$). In addition the full re-meshing will break the group definition as well as semantics definition ($f^{\ominus}$, $g^{\ominus}$).

Boolean operation on free-form solids is also presented in [15], authors use an intersection technique on the control-meshes of the two solids. Figure 2.1.a shows the meshes that have to be intersected. The main process is to compute the intersection curve between the two control meshes and then subdivide the meshes to adapt the intersection nodes. From their results, the surface rendering (fig.2.1.b) is enough good but from the mesh quality point of view (fig.2.1.c) the modified mesh is not really appropriate for the FEA because there are many degenerated and skinny triangles ($c^{\ominus}$).

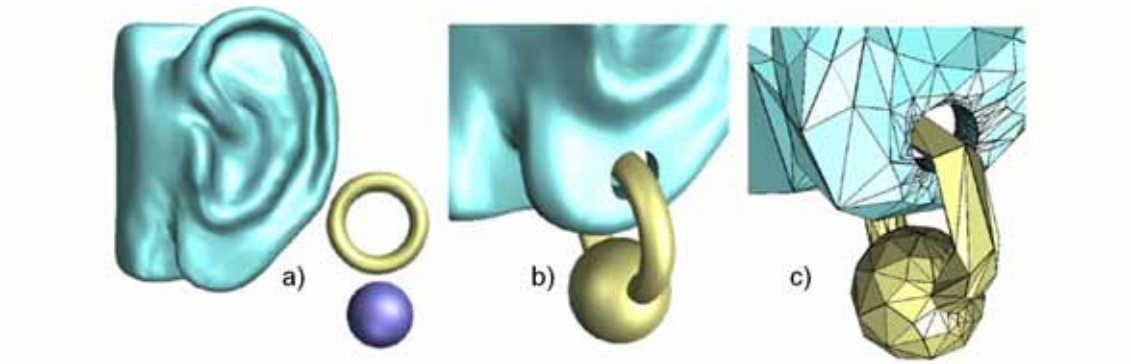


Figure 2.1: Approximate Boolean operations on free-form triangle meshes [15]

The paper [121] presents an approach for approximate Boolean operation of freeform solids with a great number of polygons in short time. The author proposes to compute Boolean operation on the mesh elements sampling schema which necessitates less time than Boolean operation directly on the mesh elements. This is useful for the FEA meshes which has usually high density. This approach is

interesting since it manages tangential contact between two operand meshes. In figure 2.2, the mesh B should be added onto the mesh A (fig.2.2.a) and the resulting mesh is A' (fig.2.2.b). The two models' boundary are somewhere intersecting and tangentially contacting. In the zoomed view (fig.2.2.c) of the resulting mesh, there are some degenerated triangles that can be identified. The result is acceptable for rendering but not really satisfactory for a FEA ($c^{\ominus}$).

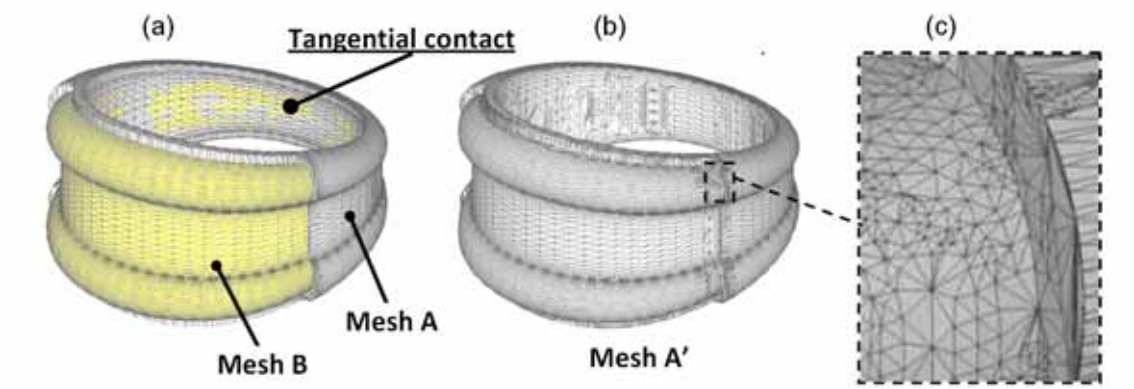


Figure 2.2: Approximate boolean operations on large polyhedral solids with partial mesh reconstruction [121]

The Boolean operation for polygonal meshes presented in [86] is computed on the adaptive and refined volumetric octree data. For intersection zone of the octree elements, the relative mesh has been transformed by inserting and swapping edges so that the mesh elements align with the intersecting octree elements. Figure 2.3.a to e show how the mesh has been transformed to align the octree cells. In figure 2.3.a the initial mesh is shown with red segments whereas the adaptive octree cells are shown by square case. The Boolean operation has been applied firstly on the octree structures of two meshes so that the gray octree elements are chosen to be kept. Therefore, the two meshes should be transformed to align the gray octree elements for matching the Boolean operation. Figure 2.3.b shows new segments (green) and points (yellow) added on the mesh. Figure 2.3.c shows the initial edges (dashed green line) that are swapped into edges (horizontal green edges) connecting the two newly inserted nodes. Then the mesh elements outside the gray octree cells are removed (fig.2.3.d) and the other mesh are also transformed at the same time. For being fully compatible, some more

points are inserted (fig.2.3.e). With this approach an experimentation result is shown (fig.2.3.f) on two initial meshes (upper and lower) having different sizes. After the Boolean operation the lower part has many skinny triangles which is due to lack of transition band ($c^{\ominus}$). In addition, with this method if the two initial meshes are positioned in a way that the newly inserted nodes (yellow) and the original nodes (brown) are very close, the produced mesh elements close to the insertion/swapping will be very degenerated. By the way, the modification performed on intersection of the two parts stays very locally ($a^{\ominus}$).

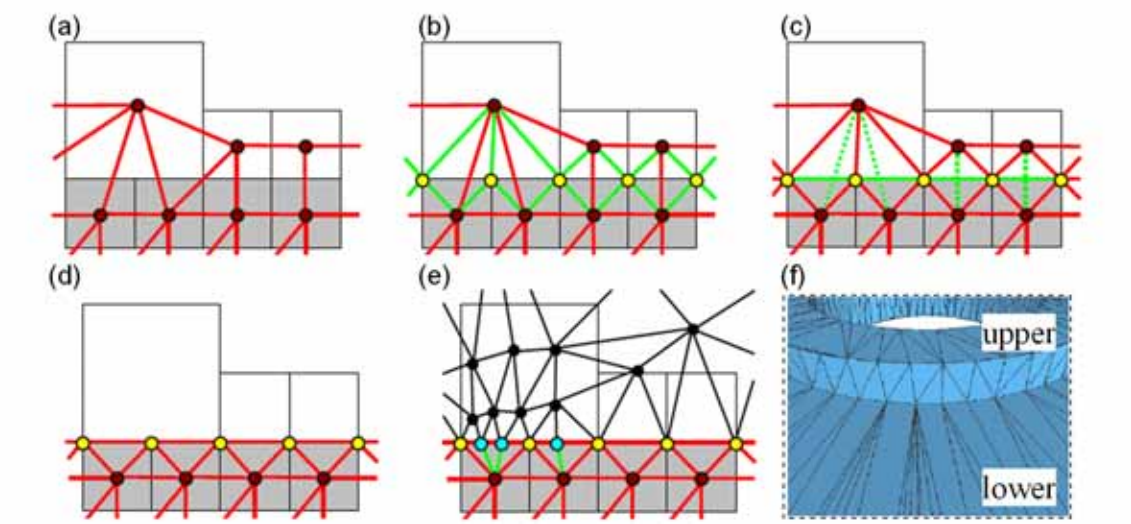


Figure 2.3: Hybrid Booleans [86]

Polygon mesh representation is also used to sketch animated 3D freeform objects [48, 49]. Boolean operation is applied to the model and the visualised mesh updated by partially re-meshing process. The construction and modification of the model is based on the strokes imposed visually by the user. The Boolean operation is applied to the strokes and the representing mesh is updated after.

Relatively to the intersection of triangle meshes, *Smith et al.* worked on a topologically robust algorithm for Boolean operations applied to polyhedral boundary models [107]. Their hierarchical decomposition of Boolean operation from solid until vertex allows avoiding topology connectivity problem due to numerical accuracy that may happen usually with other methods. It seems that the criteria a and b are difficultly satisfied because of approximate manipulations

of the user ($a^{\ominus}$, $b^{\ominus}$).

The mesh merging is usually used to render two intersecting meshes compatible in the sense that they are manifold and correctly connected. The paper [20] presents an approach for computing self-intersection of polygonal meshes which is can be interesting for evaluating the mesh quality. The paper [112] presents a fast algorithm for detecting intersection/collision between triangles on meshes which can be useful for identifying the intersection interface for merging two intersecting meshes.

Graysmith and al. have proposed a method to join two meshes [39]. The algorithm consists in four steps: contact detection, shell construction, element creation within the shell and mesh assembly. However, their method is not adapted to the treatment of meshes of heterogeneous sizes and the quality of elements in contact zone does not fit the FE requirements ($c^{\ominus}$).

Lira et al. worked on the computation of potentially multiple intersections between two intersecting meshes [27, 62]. The modifications solely affect the elements directly involved in the intersection. Thus, degenerated triangles, i.e. triangles defined by one or two small angles, may be created ($c^{\ominus}$). This is not acceptable for FEA. Figure 2.4 gives some results taken from the original article. Figure 2.4.a shows the intersection of mesh elements (originally quadrangles) identified and split into triangles with respect to intersection points. Figure 2.4.b shows that the mesh is relaxed around the intersection zone. The direct splitting of the intersection mesh elements could not be acceptable if the two meshes to merge have different densities. The splitting will produce topologically degenerated triangles and the relaxation of the meshes will not improve it (c). Figure 2.4.c presents a result of mesh merging obtained for two 2D meshes having homogenous size.

The approach of *Coelho et al.* is quite similar to the mesh merging approach proposed in this thesis (section 6.2) in the sense that it detects and removes the intersecting faces before triangulating and smoothing the created holes [27]. Again, this is not adapted to the merging of meshes having heterogeneous triangle sizes ($c^{\ominus}$). Even if it works on FE meshes, the semantic information are not taken into account as constraints for triangulation algorithm ($f^{\ominus}$, $g^{\ominus}$).

The work by *Jung et al.* focuses on the detection and removal of self-intersections

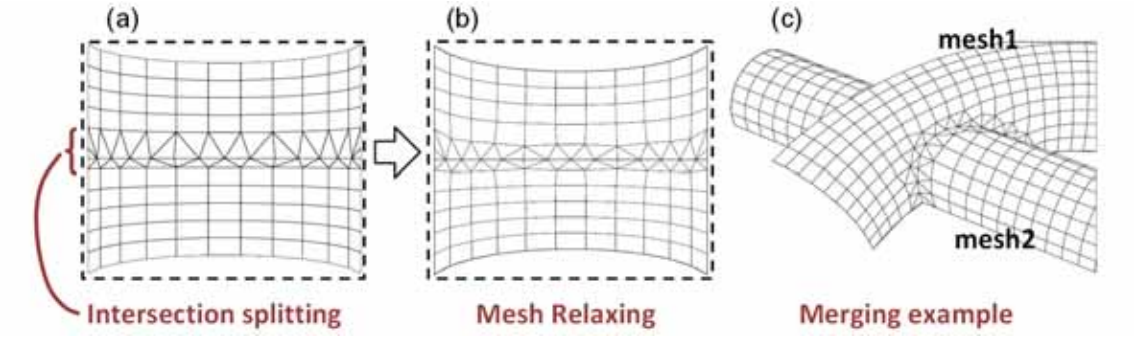


Figure 2.4: Intersecting and trimming parametric meshes on Finite-Element shells[27]

in triangle meshes during the mesh offsetting process [53]. Figure 2.5.a illustrates an example on a triangle mesh. The offsetting stage is shown in figure 2.5.b and the red region is not valid because a portion of mesh goes inside the model and the mesh elements intersecting with each others. The invalid region correction process is followed (fig.2.5.c) for removing the penetrating mesh elements and for correcting the intersection. As shown in the zoomed figure 2.5.d, the triangles directly associating with the intersection curve edges are split directly to remove the self-intersections. Therefore, the quality of the triangles produced by direct splitting is not suitable to perform the FEA ($a^{\oplus}$, $c^{\ominus}$).

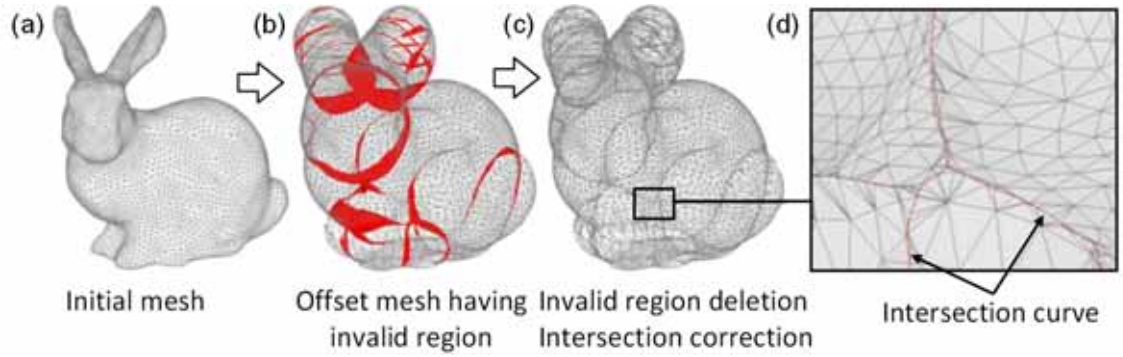


Figure 2.5: Mesh offsetting and intersection repairing [53]

Park has also been working on the efficient computation of self-intersections without however removing them [85].

The workflow developed by *Shostko et al.* is also quite similar to the one

proposed in this thesis since it computes the intersection lines, removes the intersecting faces and remeshes the created holes using an advancing front mesh generation method [104]. Figure 2.6 gives an example of a surface triangulation over two intersecting cylinders. In the zoomed picture many degenerated triangles can be identified and the modification stays local ($a^{\oplus}$, $c^{\ominus}$). The main difference between this work and the one proposed in the present thesis (section 6.2) is firstly that this method does not really optimise the intersection curve by moving/deleting the intersection nodes. Secondly, only the triangles associated to the intersection curve are deleted, and therefore the remeshing space could not be adapted to the different relative position nor to the difference of mesh density.

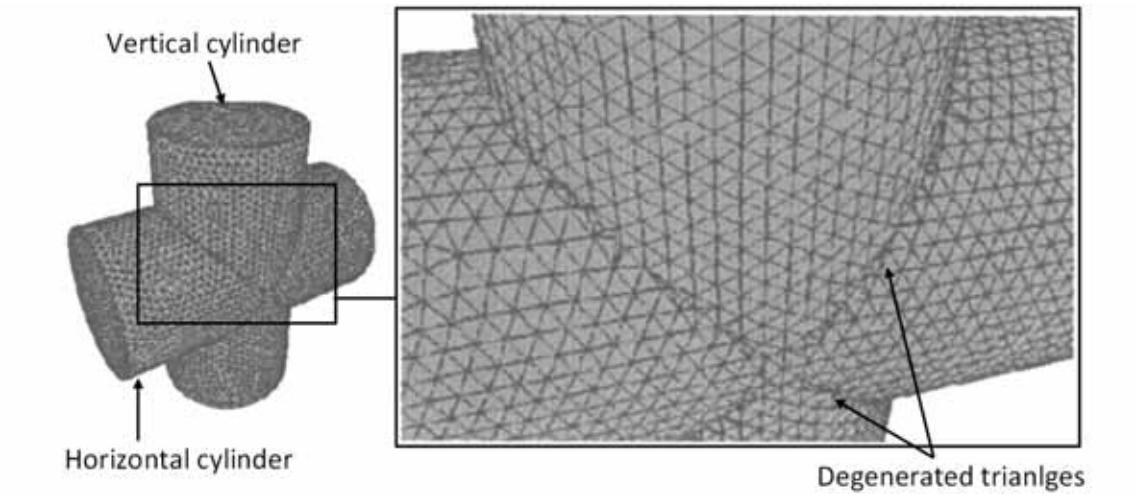


Figure 2.6: Surface triangulation over intersecting geometries [104]

Park proposed an approach for computation of polygonal extrusions. It is assumed that a 2D sweeping polygon moves in space while its containing plane is kept orthogonal to the tangent direction of the trajectory curve, a planar polygonal chain having no self-intersections [84]. *Wang and al.* have also worked on a 3D mesh extrusion method for intuitive geometric modelling of free-form polygonal models [120]. These approaches are interesting since they bypass the CAD modelling step while working directly on discrete representations. Hence, it can help the direct definition of basic primitives (e.g. stiffeners, ribs) that could then be used as inputs of mesh merging algorithm. The paper [52] has also proposed a blending method that merges several triangle meshes. This approach gives

good results according to visualisation criteria. Unfortunately, the fact that the transition surface is hardly controllable reduces its applicability to the industrial context.

Chouadria et al. worked on the treatment of contact interfaces to simplify polyhedral assemblies [23]. The faces between interacting objects are identified and specific operators are applied to merge the two parts. However, the shape of their triangles does not have to be equilateral since their models are dedicated to the simulation of assembly/disassembly steps ($a^{\oplus}$, $c^{\ominus}$). Figure 2.7 shows the re-meshing result on two models to be assembled.

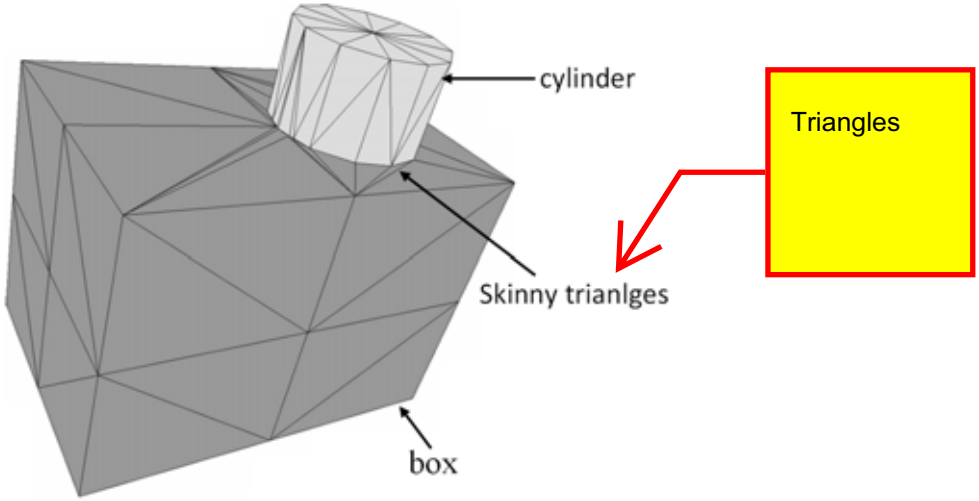


Figure 2.7: Contact interface re-meshing in context of assembly collision detection [23]

Actually, all these merging approaches are close to the face/face merging mode. *Rose and al.* have been working on manipulation operators for FE modelling in the automotive industry [95]. They deal with the deformation of FE meshes during the preliminary design phases. Unfortunately, the semantic data associated to the geometric models are not considered as additional inputs constraining the mesh deformation algorithm.

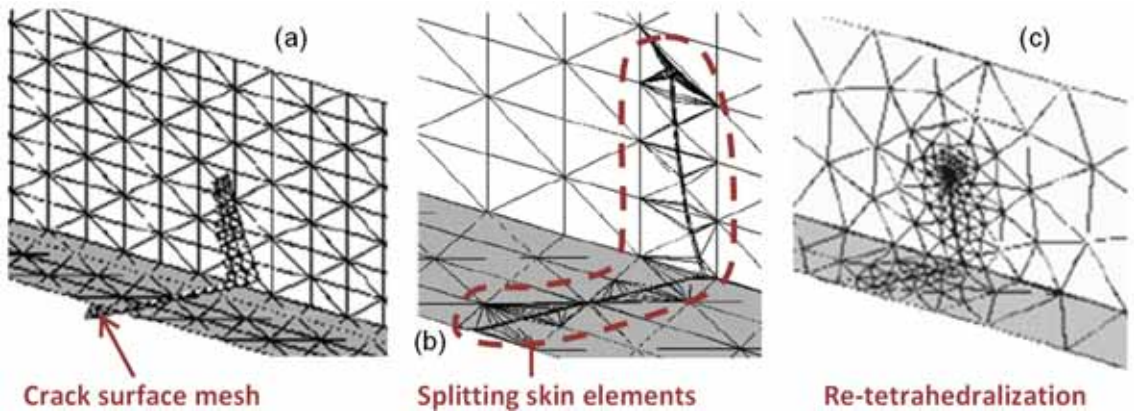


Figure 2.8: Automatic crack-insertion for arbitrary crack growth [18]

2.3 Crack feature / contact zone insertion

Bremberg and Dhondt [18] propose an approach for crack insertion into a mesh by creating a blending between the surface mesh of the crack profile and the mesh of the cracked volume. The blending needs a re-meshing process for correcting the intersection between the crack profile surface mesh and the surface mesh from the cracked volume mesh. The last is for ensuring the shape of the modeled damaged structure. It is necessary for making two sub-parts in the "cracked" mesh with respect to the crack profile. This approach requires the modelling of the crack as a mesh feature. The main disadvantage of this method is that the re-meshing process could be difficult. An example is shown in the article. Firstly, the intersection between the crack profile surface mesh and the skin of the volume cracked mesh is computed (fig.2.8.a). All intersecting triangles on the skin of the cracked mesh are split respecting to the crack surface mesh (fig.2.8.b). Secondly, the skin of the crack model combined with the crack profile surface mesh gives a closed surface mesh to be tetrahedralised (fig.2.8.c). This approach is quite efficient at step of direct splitting of hull triangles mesh but the quality of the triangles is not good for FE simulation, a lot of skinny triangles are shown in figure 2.8.b ($c^{\ominus}$). Moreover, the approach requires re-meshing process for the whole mesh. It could be very complicated according to the model's complexity and it modifies completely the tessellation of the whole mesh which is not at all appropriated for mechanically tuned meshes ($a^{\oplus}$). To improve the mesh quality

and avoid skinny triangles, skinny triangles are relaxed so that the final mesh (fig.2.8.c) has a different tessellation from the initial mesh (fig.2.8.a).

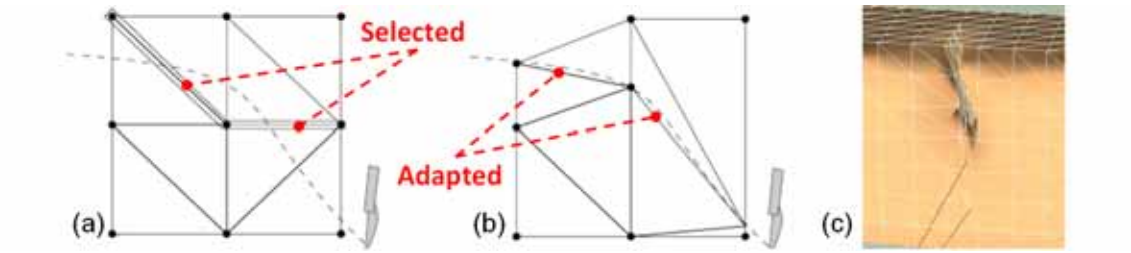


Figure 2.9: Supporting cuts and FE deformation in interactive surgery simulation [82]

Mechanical crack introduction and propagation schemes were augmented in an elegant manner with a X-FEM (eXtended Finite Element) method [10, 11, 78, 115]. The use of special enrichment functions as well as a discontinuous function along the sides of the crack allows one to achieve "irtually" a complete crack analysis (on mechanical computation level only using so-called "level sets") without any geometric mesh modification [78]. These works aim at predicting crack behaviour and not to insert a specific crack shape on the mesh without using CAD model. However, the application of this method could be difficult in the case of geometrically complex cracks having no regular shape (because it is difficult to describe such a crack feature).

In [80, 100], the insertion of cracks into a mesh model is based on the splitting of mesh elements. The proposed direct splitting of the elements is mainly aimed at real-time visualisation of cracking process. Whereas, from the FEA point of view, the resulting mesh is not appropriate because the split elements could have bad quality ($c^{\ominus}$). The use of Boolean intersection and cut operations between the original model and crack masks have been presented in [77].

Nienhuys and al. [82] describe a cutting algorithm which continuously deforms tetrahedra so that the cutting trajectory aligns with the tetrahedron face or edge. This method reduces the need to introduce new nodes but can produce degenerate tetrahedra ($c^{\ominus}$). Figure 2.9.a shows how the edges are selected according to a trajectory of cut which are presented by the dashed line. Figure 2.9.b shows how these selected edges are deformed for adapting to the cut. Figure 2.9.c shows an

experimental result of the algorithm.

The approach proposed in [56] allows multiple consecutive incisions of tetrahedra in the crack zone. In this method, a tetrahedron maintains its state information including the number and position of cuts. Multiple cuts are merged, and the affected tetrahedra are subdivided along the cut plane when a portion of the mesh is completely severed from the rest.

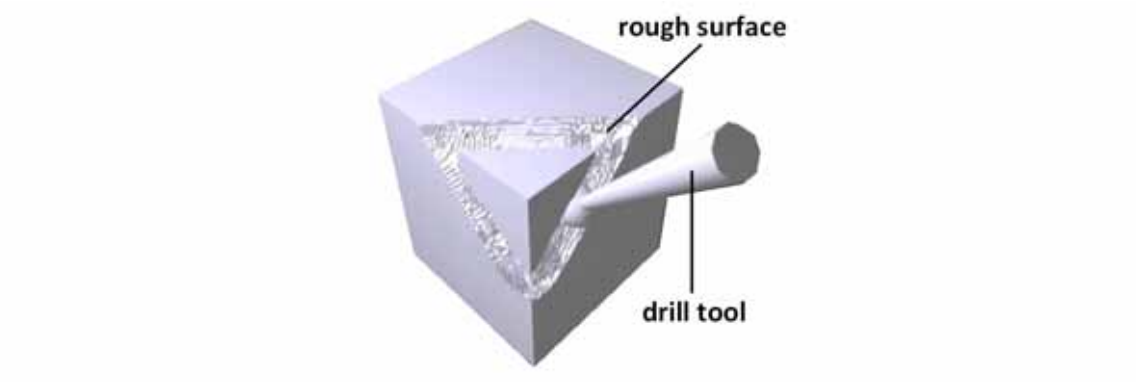


Figure 2.10: Simulating Drilling on tetrahedral meshes [116]

2.4 Cutting operation

In [14, 19, 37, 117], a cutting mesh approach is proposed which directly split the elements in order to follow the cutting trajectory. However, also in this case the mesh quality is not acceptable for FEA ($c^{\ominus}$). On the contrary, the cutting operators proposed by [51, 116] take care about the mesh quality, but the cutting profile on the mesh does not perfectly match the cutting tool ($e^{\ominus}$).

In [116], the author's idea is to create alternative representation of tetrahedron where it is partially intersecting with the drill tool. By subdividing sufficiently these tetrahedra and by removing all drill tool intersection tetrahedra, the authors obtain the mesh model with a cut shape that corresponds almost to the drill tool. Here, there exists a tolerance between the shape of the drill tool and the cut part on the mesh model. The objective is to model drilling operation in real-time on a tetrahedral mesh in the context of rendering. As shown in figure 2.10, the cut

surface obtained is very rough and the method is therefore not really adapted to industrial needs.

The paper [30] presents a method used to simulate the cutting of a mesh with a meshes tool. The intersection points between the two interacting meshes are inserted so that the mesh elements are directly split and removed. The cut tool should be meshed and the quality of the resulting mesh is not enough good for the FE simulation ($c^{\ominus}$). Figure 2.11 shows an example of cutting using their proposed method. Figure 2.11.a shows the initial mesh that will be cut by the knife surface. The resulting of the cut is shown in figure 2.11.b. The top portion is removed and the cutting facet corresponds to the knife plane. In this picture, there are many degenerated triangles appearing on the cutting facet. The proposed method insure that the modification is local which maintain the validity of the mesh tuning ($a^{\oplus}$)

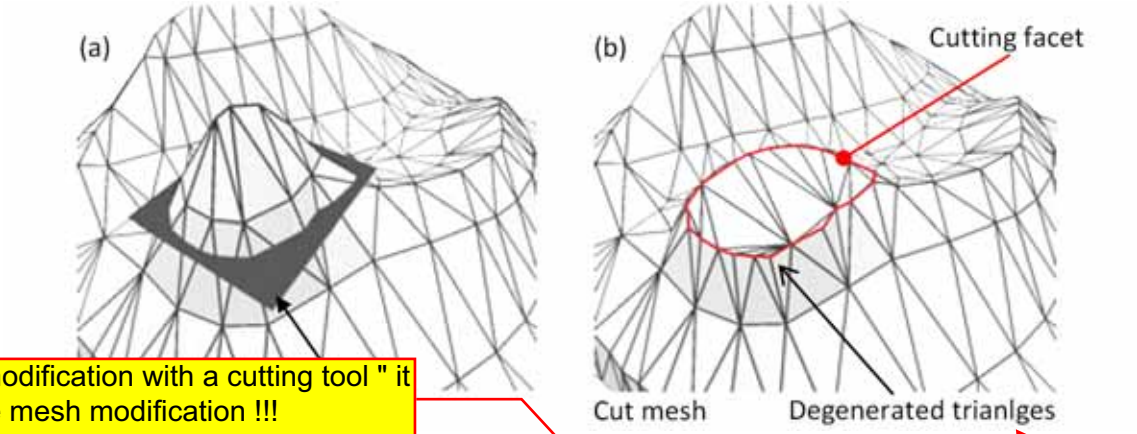


Figure 2.11: Interactive cutting on tetrahedral meshes [30]

Cutting a triangulated mesh model by a stroke on a projected screen is achieved by cutting the individual triangles of the model by the stroke. The work [79], using hand-drawn strokes to design 3D models, presents an algorithm for cutting a model by a stroke. The mesh elements are subdivided according to the strokes, the strokes are simplified: points on the strokes are either removed or replaced by nodes on the mesh. In figures 2.12.a and 2.12.b the overview of the proposed system is shown. On the screen, the user gives a stroke and a cutter surface mesh is generated on the depth direction for cutting the mesh. One cut-

ting schema is shown in figures 2.12.c and 2.12.d. The stroke is simplified and the mesh nodes are moved for adapting to the stroke. There are many degenerate triangles produced that cannot be accepted for FEA. This paper gives a simple and robust approach to cut the mesh model but the precision and the mesh quality is not emphasised ($c^{\ominus}$).

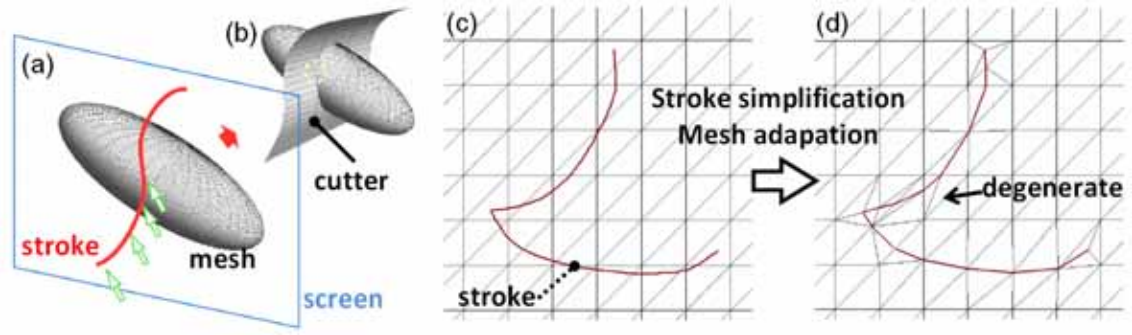


Figure 2.12: Method for cutting a mesh model by a hand-drawn stroke [79]

Nealen et al. [81] moved the vertices that lie near a stroke so that they lie on the stroke. After this, other vertices are moved by using a relaxation operation to improve triangle shapes ($c^{\oplus}$). This is a reasonable approach taking care of the precision and mesh quality.

2.5 Mesh filleting

In mechanical engineering, a fillet is a concave blending of an interior corner of a part design and it is used to avoid/reduce a phenomenon of stress concentration. Similar to the filleting, a rounding of an exterior corner is called a "round". When associating two objects an operation similar to filleting is also used to create a transition surface in between for smoothly connecting them, which is called model blending.

Blending surfaces in solid modelling are largely studied, for example, by *Rossignac* in [96, 97]. The function defining the blending surface is constructed from functions defining the surfaces to be blended with help of a piecewise quadric function [43].

The mesh blending is also studied. The direct specification of blends over a mesh, or between two meshes, has been addressed in [66]. The method uses a rolling ball blending technique for shaping triangles meshes around sharp corner by using an intermediate parameterised surface. In the context of maintenance, it could help the direct specification of blends representing welding cords. For deforming the mesh according to the intermediate surface the process needs user to specify two points for splitting the surface. This method presented in the paper [66] is mainly addressed for case where the sharp edges generate a loop whereas for mechanical engineering the filleting can applied to a sharp edge set with zero loop or more than one loop.

The surface blending which combines mesh morphing with deformation theory from continuum mechanics is introduced in [46]. From the source shape until the target one, multiple intermediate shapes are obtained by deforming the intermediate generated mesh. The deformation process follows a quasi-physical law which assigns material properties to a geometric object and uses its strain energy to measure and to minimise its distortion during the morphing process.

The surface mesh offsetting approach for scaling up the model is presented in [54], which allows producing rounded feature on the convex sharp corner. The presented method consists in offsetting nodes with the multiple normal vectors and creating a blend surface for filling a gap. The gap/overlap will be produced on the convex/concave sharp edges when moving the faces along their normal vector. In case of convex sharpness, the nodes and edges on the sharp edges will be duplicated several times so that the variation of the vectors from the node on the initial mesh to the duplicated nodes on the offset mesh is smaller than a threshold. The blend surface is created with these duplicated nodes and is an approximated conic surface. Some experimental results are shown in the article (fig.2.13). The offset mesh has a bigger size than the initial mesh and different convex sharp edges are rounded in the offset mesh. The topology connectivity around the concave sharp corner is not anymore valid after offsetting. The numbers of edges at two sides of concave sharp feature are not equal (fig.2.13.b). This approach is useful for rapid prototyping and numerical control machining for triangular mesh but could not be really usable for local FEA mesh modification for the following reasons: 1) the offsetting is moving all mesh elements which is not a local

modification ($a^{\ominus}$); 2) many skinny triangles are generated for the blend surface as the subdivision is down in the direction ($c^{\ominus}$); 3) the concave sharp corner becomes non-connected topologically or produces intersection after offsetting ($d^{\ominus}$).

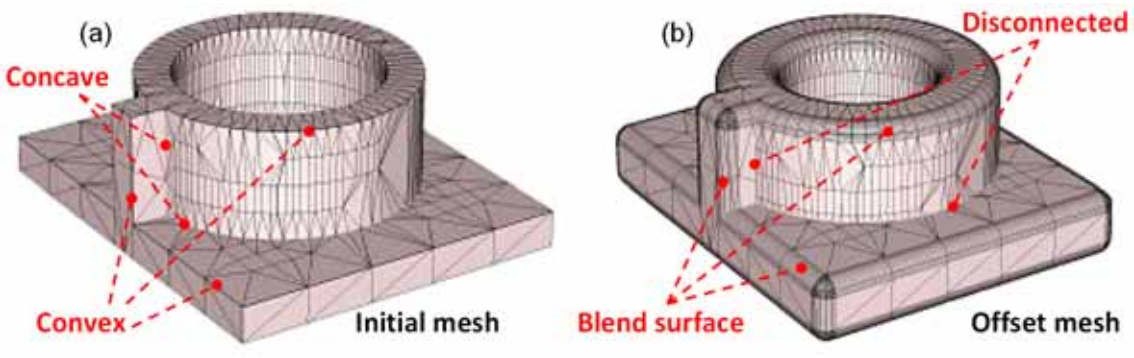


Figure 2.13: Offset triangular mesh using the multiple normal vectors of a vertex [54]

2.6 Semantics manipulations

Semantic aspects have been widely detailed and exemplified in the Aim@Shape European Network of Excellence [3] and in Focus K3D European project [35].

In industrial design, the semantic data correspond to all the information that is used to design and manufacture a product: its colors, its material, its decomposition into meaningful areas. In the context of FE simulation, they may also correspond to all the data required before running the simulation properly saying: BCs, geometric parameters (thickness of a group of faces, section parameters for beams), materials and so on [5]. Actually, semantic aspects can be encountered in all the steps of the product lifecycle. Integration and maintenance of semantic information during the product modelling process has been the subject of research since many years [13, 33] . Some approaches try to take into account this application-dependent information during the manipulation and modification of the underlying geometric models. *Hamri et al.* [41, 42] have proposed an unified framework to handle and process the CAD models and FE meshes through an intermediate polyhedral representation. In their approach, the semantic data are taken into account through the specification of partitions whose boundaries

may drive a polyhedral simplification method used to adapt the digital mock-up to the various engineering needs (e.g. visualisation, FE simulation, clash detection). This is an interesting example on how semantic information can constrain geometric manipulations.

Semantic description has also been taken into account for virtual architecture modelling. In [31, 32] the architecture semantics for different geometry components of a model are attached into the visualisation. The semantics are also defined according to multi-resolution. A part of a geometry corresponding to a semantic could be subdivided into several sub-parts standing for more detailed semantics. The ancient urban designing system based on the semantics has been also talked about [64, 65]. Instead of modelling geometrically the objects such as points, lines, etc. the system needs user to create directly urban units (ex. streets, houses, etc.). All these works use a XML file for describing the semantics in addition to the geometry file.

For doing FEA the geometric model and semantics are provided separately to a FEA solver. For example, the ASTER solver, industrial FE code using for analysis of structures ASTER [26] takes as input a FE geometry file generated from mesh software (ex. I-DEAS or SALOME [99]) and a corresponding command file containing also a physical semantic description of the mechanical modelling (modelling definition, geometric/mechanical parameters, materials, BCs, etc.). The proposal in [4] introduces a notion of "semantic CSG tree". The objective is to incorporate physical property semantics into a CSG representation of solids and to provide semantic constraints for manipulating these properties. Preserving the semantic information at the nodes of a CSG tree also guides the designer to reuse the analysis of the substructures of that tree.

2.7 Conclusion

In summary, much work has been done in terms of mesh modification and semantics manipulation. Some of them have been analysed in this state-of-the-art. However, very few approaches satisfy at the same time various criteria for direct FE mesh modification proposed in section 2.1.

Some approaches are not restricted to local mesh modifications and a complete

higher dimension re-meshing is following a lower dimension modification. More concretely, a hull surface mesh is modified and volume elements have been re-generated completely under the modified skin (ex.[18, 55], etc.). This is not acceptable when manipulating tuned FE meshes.

Some mesh modification methods do not care about the mesh quality. The aspect ratio of the mesh elements could be very bad due to direct subdividing (ex. [15, 82, 86] etc.). Bad aspect ratio of mesh elements may be produced in case the two operand meshes have similar too heterogeneous sizes (ex. [27, 62] etc.). The topology consistence may be not insured in some works (ex. [54]).

During the mesh modification, the shape of the operating tool or modification interface is not always respected. Rough modifications are proposed for visual simulation (ex. [116]). In our context, more accuracy is mandatory.

Concerning the semantics transfer, most of the works do not take into account the semantic information. There are many works and projects working on semantics enrichment of geometry models by adding annotation and using ontology (ex. [32, 65] etc.). This allows easily querying geometry models according to certain semantic information. Very few works propose to update the enriched semantics once the geometry model is modified and how to reuse the previously defined semantics. Some approaches think to integrate the BCs and the FEA results into the solid construction tree leaves for each object element (ex. [4]). This allows reusing the semantics (BCs and FEA results) once the concerned geometry elements are added into another component. However, the modification of a component is never constrained by the semantics and the semantics cannot be reused once the geometry elements have been modified. This is notably what is done in this thesis.

Actually, from the above descriptions, this thesis aims at proposing a set of models, methods and tools to modify directly enriched FE meshes. That's to say, both the geometric information and semantics are used to perform those modifications and updated right after.

PART II
CAD-less geometric modelling operators

In this part, the proposed CAD-less modelling operators are introduced and detailed. A generalised definition of mesh modification operators is given specifying the constituting components which respectively take care of the geometric operations, the treatment of mesh groups presenting in the FE mesh to be manipulated as well as the propagation of the semantic information required for FE simulation.

Several instances of the generalised CAD-less geometric modelling operator have been proposed, prototyped and tested on different mesh models.

Chapter 3

Generic CAD-less approach for fast preparing of FE meshes

This chapter focuses on the proposed approach for the direct manipulation of semantically enriched FE mesh models. The various layers of information entailed on an enriched mesh are firstly presented. To be able to manipulate the FE mesh as well as the associated information, the needs for special treatments for each layer are discussed. At the end, the global structure of the proposed CAD-less operators is shown specifying the constituting components for manipulating separately the different layers. It is worth to mention that the proposed modelling tools can be applied to meshes enriched with any semantic data, even if here the focus is on FE meshes thus considering the semantics and requirements of this specific application.

3.1 Multi-layered framework

A Finite Element Method (**FEM**) model is used to simulate "virtually" a product's physical behaviour. To perform the FE simulation, different data are required. Several information levels can be distinguished to classify these data (fig.3.1).

The geometric data can be considered as the **lowest layer** of information required for FEA (fig.3.1). This type of data describes in a discretised manner

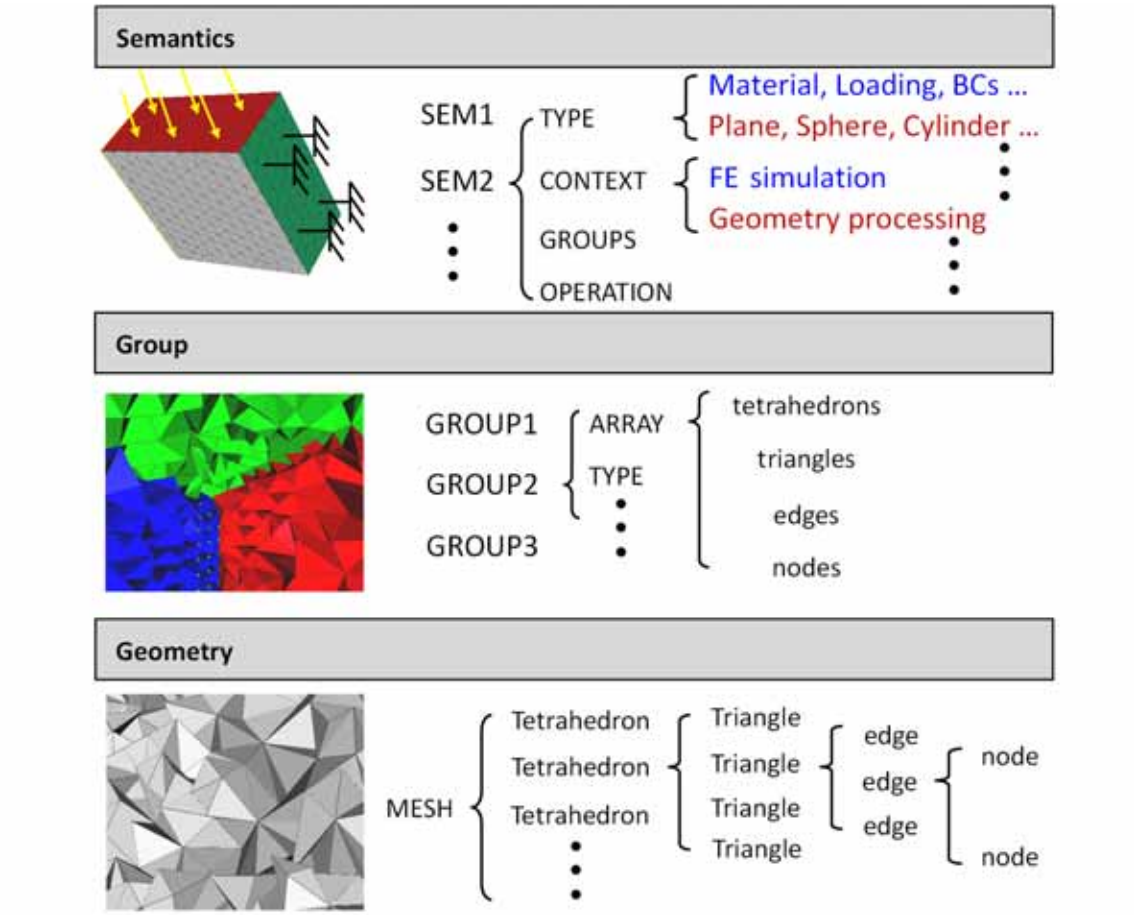


Figure 3.1: Different layers information of an enriched FE model

the shape of the structure at the level of the mesh, i.e. the object shape adapted to FE modelling. This level is consisting of simplicial complex which is a set of n-dimensional (n=0,...,3) simplices such as vertices, edges, faces (triangles, quadrangles $\cdots$) and 3D finite elements (tetrahedra, hexahedra $\cdots$).

At the **highest layer** (fig.3.1), depending on the type of simulation, different kinds of semantics are involved. The semantics defined on the geometric model is used to enrich the model with certain information considered as input parameters for simulation:

- To cover the various FEA needs, different kinds of physical semantics are necessary to simulate the various behaviours (mechanical, thermal, fluid interaction, etc.).

- To apply some animation such as assembly simulation, different semantics relative to the shapes might be defined on the mesh model. The shape semantics are defined on the model or its sub-parts such as plane, sphere, cylinder, etc.
- To answer other needs, other semantic information will be enriched on the mesh model.

In order to relate such semantic data to the geometric model, an additional **intermediate layer** is needed (fig.3.1), which is constituted by groups (sets) of FE entities of different dimension (volumes, faces, edges, nodes). Groups are used to support different semantic data for FEA, shape description and so on. For example a group of tetrahedra representing a part of a 3D model can be used to specify a particular material behaviour law to simulate locally a complex behaviour of the structure. A group of triangles can be used to apply a pressure or to describe a contact interface between two components, etc. It could also aggregate faces staying on a sphere or other shapes. A group of edges may idealise a beam-like part or may localize a sharp feature. A group of nodes can be used to apply nodal BCs (forces, displacements) or define punctual mechanical features (stiffness, mass, $\cdots$). FE groups can be considered as low-level semantics of geometric nature.

During the direct manipulations of FE meshes enriched with different FE group/simulation semantic information, the proposed CAD-less modelling operators should correctly maintain and update all geometric/physical simulation semantic data associated to a given mesh. Moreover, since the associated semantics have a direct impact on the modifications, it must be taken into account during any modification and they should be able to interact with the three layers of information, as illustrated in figure 3.1.

At the lowest layer, the "geometric" one, only the mesh model is concerned: the mesh operators allow doing a set of necessary modifications such as removing/adding mesh elements or changing coordinates of nodes. At the middle layer, called "geometric semantics" level, the proposed operators handle the groups of FE entities of different topology. At this level, the operators mainly aim at preserving groups definition during the geometry modification performed at lowest

geometry level. To achieve this, the operators detect the key elements of the FE groups and constrain the geometric modification process so that the group definition can be restored after the mesh modifications. Similarly, at the highest level, the simulation semantic data have to be handled by the operators. The semantics may concern the FE simulation semantics such as "materials", "Boundary Conditions" or shape semantics such as "plane", "sphere" and "cylinder". These data are used to set specific geometric constraints for mesh modification operators and to specify/update accurately the FE groups of the meshed structure. In other words, the proposed operators have to update correctly all the groups according to different mesh modifications. The updating may be a removal operation of semantic data from certain FE entities in the group or a propagation of simulation semantic data onto new entities created during the local re-meshing and mesh deformation stages.

3.1.1 Geometry level or mesh modification

Mesh modification operators handling 2D/3D mesh models can be classified into 4 categories: material union, material subtraction, material fissuring and material deformation. Figure 3.2 shows some mesh modification examples.

The **material union** corresponds mainly to meshes merging one mesh to another one. The two FE mesh models in the pictures figures 3.2.e and .f are merged into one mesh model presented in figure 3.2.g and ready for new FE simulation.

The **material subtraction** removes elements from a mesh. The initial mesh shown in figure 3.2.a is modified by several kinds of material subtraction operators. The Incision and Drilling mesh results are shown on the model in the picture figure 3.2.c. The incision operator uses a width parameter L and the drilling operation requires a radius R. The Cutting operation is performed for removing a part from the mesh (fig.3.2.d).

The **material fissuring** does not change the occupied volume of the mesh model but divides the given mesh into two sub-meshes and inserts overlapping elements (2D overlapping elements in case of a 3D mesh or 1D overlapping elements in case of a 2D mesh) with opposite normal in the contact zone thus

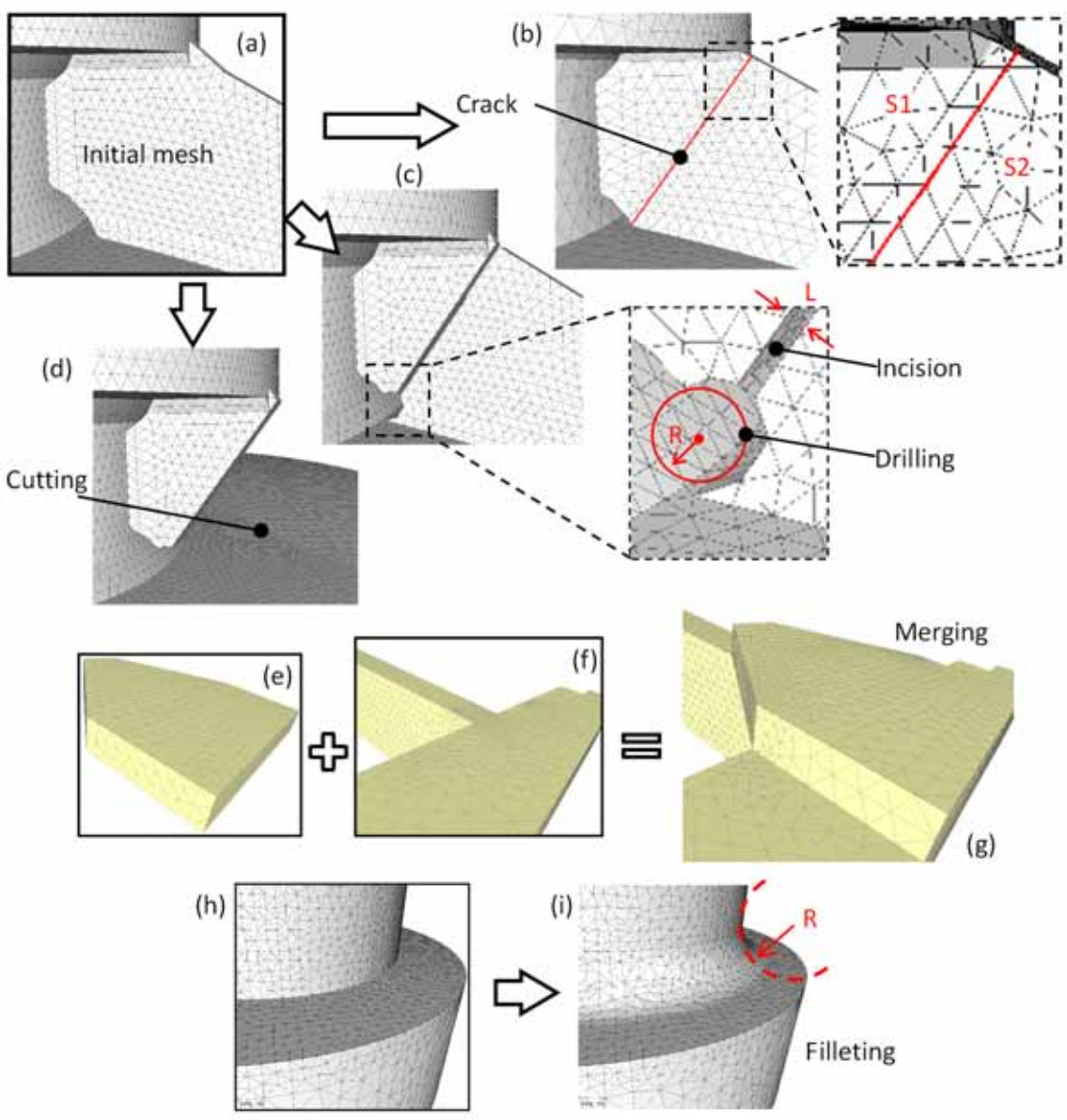


Figure 3.2: Considered geometric modifications at the lowest level of the framework

splitting the mesh model along a given surface/curve. Figure 3.2.b illustrates an example of planar fissure insertion performed on the initial mesh of figure 3.2.a. The fissuring makes the two sub-parts "S1" and "S2" separated with a planar contact interface whose corresponding nodes of the two sub-parts have the same position. The material cracking operation can be seen as introducing of a contact zone in order to simulate the interaction between two objects in contact, if in the initial mesh model such contact zone is not present.

The **material deformation** will change the model shape without adding and/or deleting mesh elements but simply modifying the nodes positions. One example of material deformation is the filleting operation for rounding a sharp corner. The sharp corner on the model shown in figure 3.2.h is filleted as illustrated in figure 3.2.i with a radius R. Such an operation can be seen as a particular operation of material adding.

Since the quality of the mesh elements strongly affects the FEA accuracy, the quality of the mesh elements should be handled during different mesh modification operations. Section 4.3 ($p_{.120}$) comes back on the quality of the mesh while considering its aspect ratio as well as the self-intersections.

3.1.2 Group or geometric semantic Level modification

Being groups composed by mesh elements, the geometric modifications may affect directly the definition of groups. At the opposite, groups may constrain the geometric modifications. If we do not take care about the group information, the geometric modifications may generate invalid groups:

- During either the topological modifications, or the re-meshing or the refinement/subdivision steps, some mesh elements may be removed or possibly substituted by new ones. When mesh elements are removed, the connections between the related groups and removed mesh elements are lost, while the newly created elements have no relation with any groups.

- During the mesh deformation that consists in repositioning nodes for achieving a certain shape. If nodes defined in a group or nodes associated with mesh elements defined in a group are moved by the deformation process,

these nodes with their new position may not be anymore valid for the concerned groups as the shape of the group is changed.

The example in figure 3.3.a presents a mesh model on which 5 surface groups (G1 to G5) and 1 node group (G6) are defined. G6 is the group containing all nodes at the top level of the part represented by the surface group G2. The considered modification consists in adding two new meshed parts on the right and left sides of the part represented by surface group G3. The added new meshes have the surface groups (L and R) respectively. Figure 3.3.b illustrates the suitable result of the mesh operation addition of the two new meshes whereas the unexpected result is shown in figure 3.3.c. In the result shown by figure 3.3.c the modifications performed on different places might be re-meshing, subdivision and deformation. The group shape for G2, G3, L and R is transformed into a strange way. Especially the frontiers of these groups are not corresponding at all to the initial ones. The white part pointed as "null" consists of new mesh elements created during the material adding operation and they do not belong to any group. The group G6 is also changed in the sense that the nodes are not anymore on the "line" as in the initial model and are not successive by adjacency.

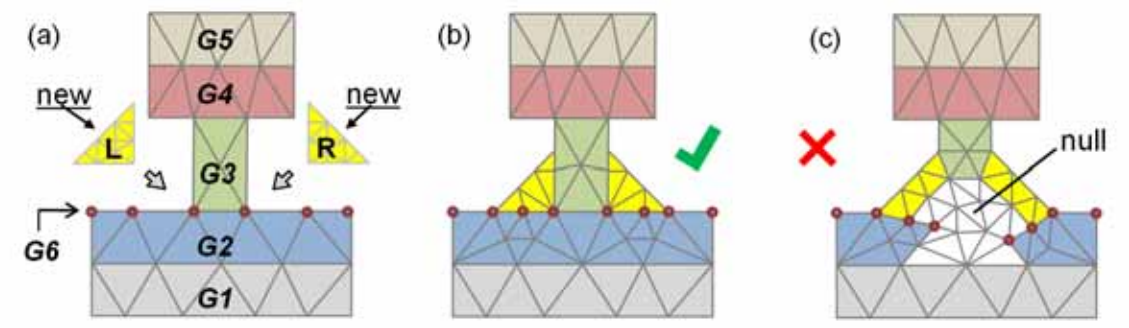


Figure 3.3: Examples of geometric modifications with (b) and without (c) group preservation

From these considerations we can identify two important aspects for preserving the groups in a meaningful way: the shape of the group and the content of the group.

The shape of a group is computed on the **space coverage** obtained by the connected elements of the group. The dimension of the coverage space varies from

0 to the mesh dimension. The dimension of group elements could be equal to or lower than the dimension of the coverage space. The dimension of the coverage is computed from the group elements and the maximum dimension adjacency. In case of tetrahedral meshes, the different coverage space can be achieved:

Given $\boldsymbol{G}$ a group of dimension n, m the dimension of the concerned mesh $\boldsymbol{M}$, $\boldsymbol{G}$ can have $(n+i)$ dimensional **C**overage **S**paces ($\boldsymbol{CS}$), where $0 \leq i \leq (m\text{-}n)$, constituted by the $(n+i)$ dimensional elements of $\boldsymbol{M}$ that have all the connected n-dimensional elements belonging to $\boldsymbol{G}$.

In other words:

- For a group of nodes:
  - the 3D coverage consists of the tetrahedra that are associated only with nodes in the group;
  - the 2D coverage consists of the triangles that are associated only with nodes in the group;
  - the 1D coverage consists of the edges that are associated only with nodes in the group;
  - the 0D coverage consists of the nodes in the group.
- For a group of edges:
  - the 3D coverage consists of the tetrahedra that are associated only with edges in the group;
  - the 2D coverage consists of the triangles that are associated only with edges in the group;
  - the 1D coverage consists of the edges in the group.
- For a group of triangles:
  - the 3D coverage consists of the tetrahedra that are associated only with triangles in the group;
  - the 2D coverage consists of the triangles in the group.
- For a group of tetrahedra:

– the 3D coverage consists of the tetrahedra in the group.

For semantics management, the coverage of maximum dimension is taken. Therefore, the coverage of lower dimension is included in the coverage of higher dimension, the coverage of lower dimension is ignored. For example, if a group of nodes has a 2D coverage, its 1D and 0D coverage is ignored. The details of space coverage, as well as of its body and boundary computation from groups are presented in chapter 5 (p.147).

The shape of a group is defined on both the body and the boundary of the coverage. Therefore, a group may present a **punctual shape** (**0D**), a **curvilinear shape** (**1D**), a **surface shape** (**2D**) or a **volume shape** (**3D**). The volume shape that is used to describe the 3D space coverage is not used in this PhD. A surface shape describes the boundary of a 3D coverage or the body of a 2D coverage. The curvilinear shape defines the boundary of a 2D coverage and the body of the 1D coverage. The punctual shape defines the boundary of a 1D coverage and the 0D coverage.

In the example shown in figure 3.3.a the group of nodes G6 has a 1D space coverage that includes the set of edges associating with two nodes in G6. Therefore the group G6 has a curvilinear shape, i.e. a straight line, and has punctual shape boundaries, i.e. the two extremities of the line.

The content of the group (body of the group) is defined topologically by the referred mesh elements. It is also equivalent to the topology of the space coverage. For example a group of faces loses a face for which there will be a hole in the group. In an example of edge groups containing a set of consecutive edges, the fact that one edge is missed from the group will let the group contain two connected edge sets. In the example of figure 3.3.c the face group G2 has lost some triangles and some new nodes are inserted between two adjacent nodes of the group G6 in comparison to the initial groups shown in figure 3.3.a. Therefore the group G2 of triangles will have two sets of connected faces and the group G6 of nodes will have several sets of adjacent nodes, thus its topology is changed.

Therefore the key point for group preservation goes through the preservation of the group coverage space. Details about group preservation are presented in the chapter 5 (p.147), where the notion of Virtual Group Boundary is presented so

that the boundary is computable for space coverage groups of all dimension. The group boundary preservation results in applying constraints in during the mesh modification such as constrained re-meshing, constrained deformation, etc. The different constraints are derived from the group boundary information and used according to the associated semantics. The group content preservation consists in re-assigning the group information on the mesh elements which are newly created / substituted by the process of re-meshing.

3.1.3 Simulation semantic level modification

The simulation semantics on a mesh model correspond to simulation type information: different FE data, shape of the associated groups, etc. Being the semantic information defined on a mesh model supported by the mesh element groups, the modification of the mesh geometry may have different effects on the attached semantics. It may be not affected, it can lead to be non-valid anymore, or it may refer also to additional mesh elements. The transfer of simulation semantics could be realised by changing the semantic information property such as parameter of a shape, material, direction of forces, etc. It could be also realised only by changing the definition of the group elements that supports this semantics. Semantics transfer depends on two factors: meaning of the semantics itself and type of modification performed on the mesh shape. In the following, the main types of simulation semantics considered in the thesis are described.

3.1.3.1 Model Shape Semantics

The shape of the model is an important semantics. It should be changed by geometry modification only if it is the intention of the expert applying the modification. Figure 3.4 shows a model example on which two geometric modifications are performed. Figure 3.4.a presents the initial model on which three shape semantics of “plane” have been defined on three different face sets (groups). In figure 3.4.b the initial model has been modified by inserting a partial cylindrical hole which corresponds to an operation of material deletion. The two shape semantics “Plane.1” and “Plane.2” are describing two sub-domains of the initial model and the new shape semantics “Cylinder.1” is created for describing the

wall of the cylinder hole. Figure 3.4.c shows a material addition modification performed on the model by adding a half-sphere on the face initially associated to "Plane.2". Therefore the zone associated to the shape semantics "Plane.2" is reduced so that the new shape semantics "Sphere.1" is defined on the spherical surface part. From this example it is clear that the updating of the semantic information layer always goes with the geometric modification. Because some mesh elements are not anymore associated with existing shape semantics after modification, new shape semantics appears and associates with some mesh elements. Typically in this example the shape semantics "Plane" has less mesh elements in figure 3.4.b than initial one in figure 3.4.a. The shape semantics "cylinder" appears in figure 3.4.b.

From industrial example analysis, such semantics are generally not associated with mesh groups containing in FE mesh model. Shape semantics can be defined by the user or automatically evaluated from boundaries of mesh groups and/or from boundaries of mesh model (knowing the sense of the normal vector to the face element, for example). Shape semantics may be used to constraint the mesh modification process in order to preserve the shape of the model.

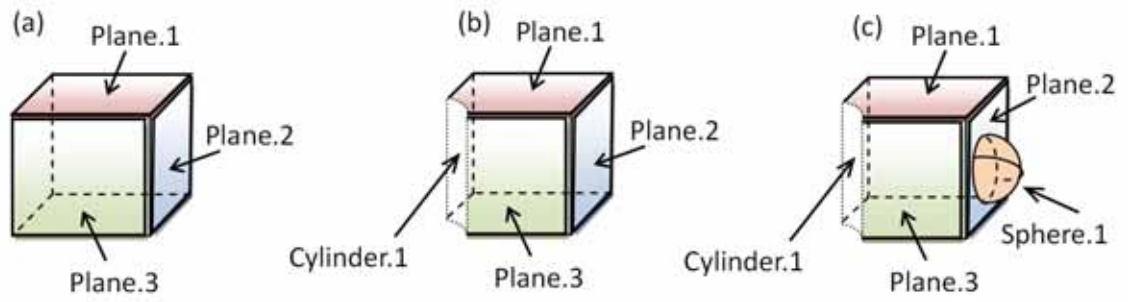


Figure 3.4: Shape semantics changes derived by geometry modification

3.1.3.2 Finite Element Analysis (FEA) Semantics

Different kinds of physical simulation semantics are needed in FEA, such as type of modelling material, thickness, stiffness, BCs, etc. These semantic data will also be transformed during the FE mesh modification process. For example, the structure can be composed from components having different material properties

representing mechanical semantics (Young's modulus, Poisson's ratio, non-linear behaviour laws). Thin structure can be idealised as a shell part needing thickness parameter (geometric nature simulation semantics) FEM model can be submitted to different type of loads like pressure (applied to a set of face elements), concentrated forces (applied to a set of nodes), displacement (imposed to given nodes or 1D/2D elements), different displacement relationships to simulate an interaction phenomenon or to handle a contact problem, etc. A more complete summary of FEA semantics and supporting mesh element's dimensions is detailed in chapter 5 (p.147).

Figure 3.5.a illustrates an initial model on which a FEA semantic, pressure, is defined on the top surface. A modification applied to the object consists in adding a half-sphere on the top surface by material addition or material deformation. Thus, several FE simulation semantic transformations are possible. In the simplest, shown in figure 3.5.b, the pressure data semantics is eliminated on the part of the planar surface covered by the half sphere. Therefore only the remaining planar surface part on the top level still receives the pressure. Figure 3.5.c shows the right result of semantic data propagation on added spherical part meshed. The difficulty is to decide how to update/transform the FEA semantics after geometric modifications: simulation semantics have to be propagated or not, how to propagate in case of specific BCs (direction of pressure, values of nodal forces, for example), etc.

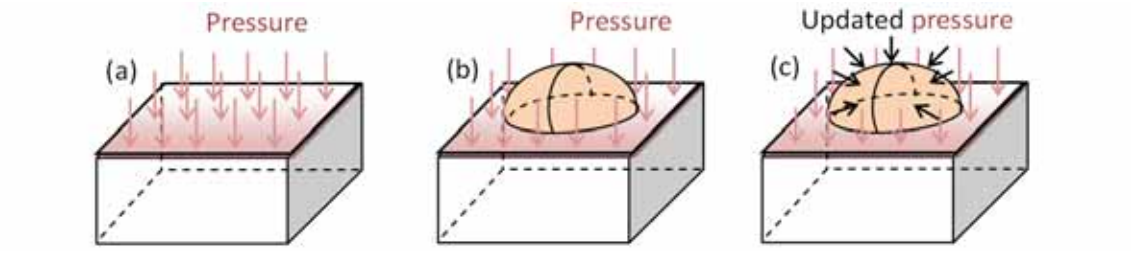


Figure 3.5: FEA semantics changes according to geometry modification

3.1.3.3 Shape of group

The shape of group is a kind of semantic information that describes the shape of the body and boundary of the space coverage for the group (see previous section

3.1.2) (p.74). The model shown in figure 3.5.a contains two groups, a surface group G1 and a volume group G2. For the group G2, its coverage boundary is a cylinder and two disks at top and at bottom (fig.3.5.b), thus the associated shape semantics provide the related surface information. The group coverage body shape is not taken into account here. For the group G1 its coverage boundary is the set of the curves bounding the rectangle as shown by the dashed line in figure 3.5.c, while its body is the rectangular planar area. Similarly for G1 the possibly associated shape semantics specifies the related planar surface and lines' information. During the geometric modifications, different constraints should be imposed from the shape semantics of groups in order to make these semantics valid all along the process.

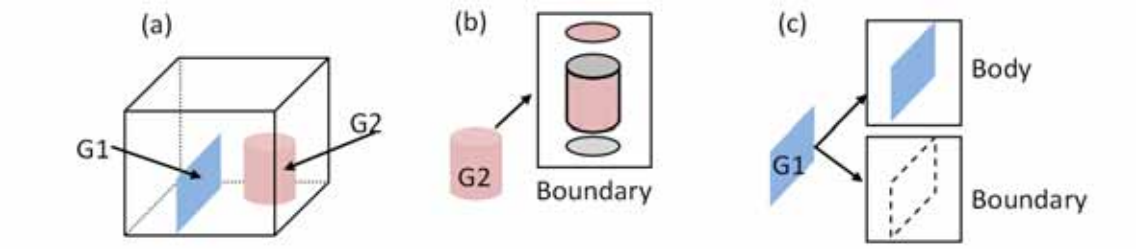


Figure 3.6: FEA semantics changes according to geometry modification

3.2 Components of the CAD-less geometric modelling operator

To take into account the multi-layer information during the FEA mesh modification this thesis proposes a generic CAD-less modelling operator concept following a component-based approach that contains various mesh treatment steps removable and re-arrangeable. Generic CAD-less geometric operator designed in this way allows reusing basic operators acting on different information layers and adding new ones. This framework eases the definition of specific enriched FE mesh modification tools taking into account different geometric and simulation semantics requirements. The generic CAD-less operator's general structure with its procedure schema is illustrated on figure 3.7. The operator takes as input a se-

mantically enriched mesh for modifying it while transferring the already existing semantics. The operator is consisting of three important stages: information exploitation, geometric modification and semantics transfer. The geometric, group and simulation semantic information are exploited at first. This information is used to constrain the mesh modification. The physical semantics and mesh groups are updated at the end. The different processes are categorised into geometric (red text) and semantic (blue text) processes. These two categories are presented separately in chapter 4 (p.85)and chapter 5 (p.147).

3.3 Information exploitation

At this stage the geometric information associated to the object are extracted. This stage should not be repeated in case several operators are performed successfully in the same system, it is performed once.

Among them, the mesh model and groups' shape are evaluated, if not already available. Specific algorithms are applied on the geometric model for detecting basic shapes possibly delimiting different object parts. The basic shapes might be plane, sphere, cylinder etc. as well as sharp edges delimiting them, which are useful for some CAD-less operator instances such as the filleting operation.

The shape recognition techniques are presented in sections 4.1 and 4.2.

The topology of the group boundary and body is also analysed (see section 5.2.2 on page 154). For simplifying the semantics transfer, the overlapping groups are treated by tools presented in section 5.2.4 (p.160) to identify portions associated to the same set of semantics.

With the shape semantic information carried by the input model the different geometric information are also identified.

3.4 Geometric modification stage

At this stage the "geometry modifications" are performed on the mesh model. It consists of topologic modifications and deformations under constraints defined from information of shape and group elaborated at the first stage. The topologic

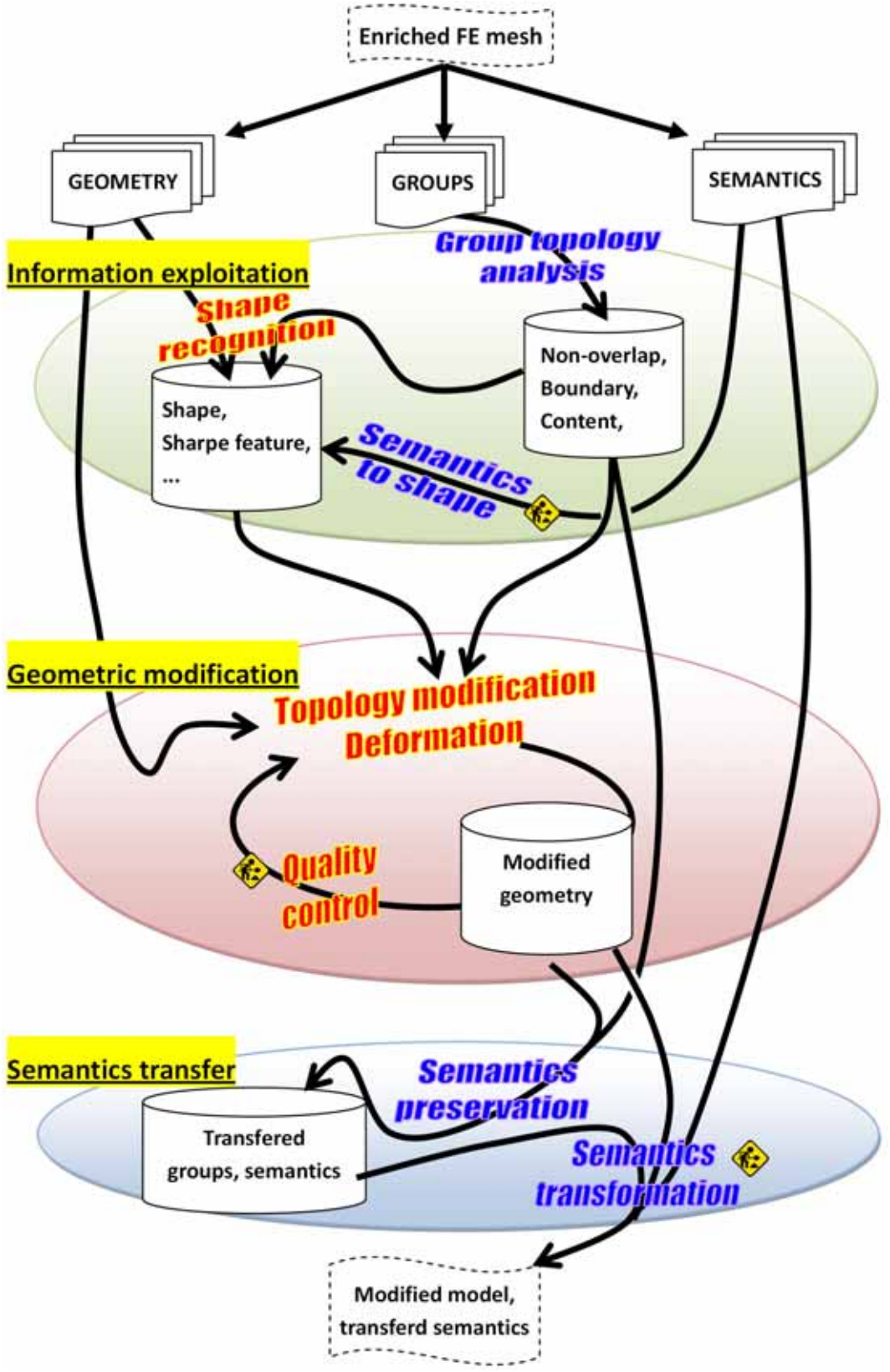


Figure 3.7: Component-based of the CAD-less mesh modelling approach

modifications concern mesh elements: deletion, addition, duplication, subdivision, etc. These modifications are detailed in section 4.4 (p.121). The deformation consists in repositioning the nodes and it aims at obtaining/maintaining a certain shape and/or at obtaining a acceptable mesh quality while relaxing the mesh. The deformation tool described in section 4.5 (p.129) allows deforming the mesh under different constraints of shape and positions. Moreover, when doing mesh deformations, we avoid topological changes and we enable local modifications.

The mesh quality control assured during the mesh modification steps corresponds to control of mesh element shape, self-intersection, etc. These aspects are presented in section 4.3 (p.120). In case if the quality is not acceptable additional topologic modifications and deformations are launched.

3.5 Simulation semantics transfer stage

At the last stage, the geometry of the input model has already been modified under different shape and group constraints. This step correctly updates the group definition of the modified or added elements (see section 5.2.3 on page 158).

The "semantics transfer" can possibly propagate the semantic information to additional mesh elements, change the associated specification value, or remove the semantics association from some mesh elements, etc. (see section 5.3 on page 166).

The proposed generic CAD-less mesh modelling approach presents a paradigm for operating on enriched FE meshes. In this thesis several instances of this paradigm have been prototyped: mesh merging, cracking, drilling and filleting. They are all detailed in the chapter 6 (p.179).

Chapter 4

Basic methods and tools for the geometric treatment of enriched FE meshes

In this chapter various methods and tools are introduced. They are used to achieve different geometric modifications included in the CAD-less mesh modelling operator introduced in the previous chapter. They involve shape recognition, sharp feature identification, and topological modification, mesh deformation under constraints and mesh quality control.

More precisely, and as illustrated on figure 3.7, the shape recognition is used to identify characteristic surfaces represented by a set of 2D mesh elements. The sharp feature identification consists in finding the mesh elements on which there are important curvature variations. The topological modification allows changing the connection of the mesh elements, such as insertion of new elements in a mesh, duplication of mesh elements for disconnecting two sub-meshes, swapping of mesh elements and so on. The mesh deformation tool allows repositioning nodes in a mesh for achieving a certain shape by relaxing the mesh elements in order to maximise the mesh quality. The mesh quality control checks the aspect ratio of the mesh elements and detects potential self-intersections.

It exists many approaches proposing purely geometric treatment of meshes. The main goal of the thesis is to set up as complete as possible CAD-less mesh

modelling operators manipulating semantically enriched mesh model under different industrial constraints. The operators includes all constitutive elements required to realise the complete loop in terms of fast industrial preparing of FE mesh model ready for simulation.

4.1 Shape recogniser

The types of shape (cylinder, sphere, plane and so on) which appear on a mesh model are not explicit since the mesh is consisting of elementary basic entities such as: triangles, quadrangles, tetrahedra and so on. Therefore, to be able to exploit such information, it is mandatory to process the geometric model and extract the shape information. The extraction of shape information will help the geometry modification to preserve a certain shape represented by some mesh elements.

4.1.1 Categorisation of basic shapes

In the proposed shape recogniser, the shape primitives to be identified correspond to four categories:

- Plane: defined by a point $\boldsymbol{P}$ and a normal $\boldsymbol{N}$,
- Sphere: defined by a point $\boldsymbol{P}$ and a radius $\boldsymbol{R}$,
- Cylinder: defined by a point $\boldsymbol{P}$, an axis vector $\boldsymbol{N}$ and a radius $\boldsymbol{R}$,
- Freeform: when the part of the mesh to be tagged does not match one of the previous types.

When considering mechanical models, there are more chances to have the first three analytical shapes instead of freeform shapes which are more common in case of industrial products. The freeform shapes are not characterised by any specific mathematical properties as in the case of the first three shapes, that is moving a point on a plane has a null displacement in one direction, the distance from one point to any point on the sphere is constant and the minimal distance of any point on the cylinder to one straight line is constant.

The various methods used to identify the shape equations approximated by a given set of triangles S are presented in the following sections. These methods [9] are based on the approach proposed in [7]. The subset of elements corresponding to the identified shape are collected and represented in the following data structure, where the list "Faces" contains the geometric elements (triangles, because we focus in the thesis on free meshes) representing the shape. P, N and R represent the parameters defining the equation of the detected surface. Each of the developed algorithms for surface shape recognition is then returning data structure elements of the corresponding type.

PLANE	SPHERE	CYLINDER
P : REAL ARRAY[3]	P : REAL ARRAY[3]	P : REAL ARRAY[3]
N : REAL ARRAY[3]	R : REAL	N : REAL ARRAY[3]
		R : REAL
Faces : LIST	Faces : LIST	Faces : LIST

Table 4.1: Data structure for shapes

4.1.2 Technique to identify planar area

In 3D space, a possible way for defining a plane is by specifying a point and a normal vector to it. To identify planes on meshes containing triangular elements, different methods could be used. In each method, a candidate plane is identified at first then triangles having a distance from it smaller than a given threshold are gathered together. The reference plane can be computed on a triangle or several triangles.

When only one triangle is used for evaluating the reference plane it is possible to use two techniques: one using the normal vector and the other using the plane equation. The first method gathers triangles whose normals are similar to the normal of the reference triangle. The second computes at first the equation of the plane "$f(x, y, z) = 0$" going through a chosen reference triangle then the neighbour triangles having deviation from the computed plane less than a threshold are gathered together. When more triangles are considered to evaluate the candidate plane the best fitting method is used. Then this method gathers all

neighbour triangles having distance to the plane less than a threshold.

In the implementation, all the triangles to be analyzed (possibly the whole mesh) are initially inserted in a set S. Each time a plane is defined the concerning triangles are removed from the set S.

4.1.2.1 The normal vector method based on one triangle

First, an initial reference triangle t is taken from the initial set S and put into the plane set P. Then, for this triangle t, its unit normal vector is computed and considered as the reference normal N_{ref}. This reference normal is then compared with the unit normal vector N of each neighbour triangle belonging to S. If the angle between the two normals is less than a given threshold ξ the neighbour triangle is put in the set P. For each triangle newly inserted in the plane set P, its neighbour triangles in S are checked. Algorithm 1 presents the pseudo-code description of this algorithm.

The main weakness of this algorithm is related to the fact that a plane is defined by a point and a normal whereas this method takes into account the normal vector only. Therefore, this method will consider two planes having similar normal and quite different points, as one same plane. Figure 4.1.a shows from a section view an example of triangle mesh, the segment between two points can be considered as triangles. The reference triangle and its corresponding normal are highlighted in red and surrounded by dashed circle. By using the normal method with a certain threshold, the number of triangles which could be considered staying on the same plane as the reference triangle, varies. If the threshold is set by a big value too many triangles might be considered as staying on the same plane (fig.4.1.b). Whereas if the threshold is very small, only very few triangles could be gathered (fig.4.1.c). While in the ideal case, for the given example, the mesh triangles should be gathered into three planes as shown in figure 4.1.d.

4.1.2.2 The planar equation method based on one triangle

In this method, the plane equation in the form of the equation $ax+by+cz+d=0$ is used. From a reference triangle, the parameters a, b, c and d are computed.

Algorithm 1 Plane identification using normal vector computed on one triangle

```
Function planeNormalMethod (S As List, ξ As Real) As Plane
 1: Variable refT, t, tt As Triangle
 2: Variable n(3) As Real
 3: Variable A, B As List
 4: Variable plane As Plane
 5: refT ← S^0
 6: removeElementFromList(refT, S)
 7: addElementToList(refT, plane.Faces)
 8: addElementToList(refT, A)
 9: plane.N ← normalOfTriangle(refT)
10: plane.P ← coordinatesOfFirstNode(refT)
11: while A ≠ Φ do
12:    /* Check the adjacency of triangles in A */
13:    for all t in A do
14:       for all tt in S do
15:          if isAdjacent(tt, t) then
16:             n ← normalOfTriangle(tt)
17:             if ‖plane.N − n‖ ≤ ξ then
18:                addElementToList(tt, plane.Faces)
19:                addElementToList(tt, B)
20:                removeElementFromList(tt, S)
21:             end if
22:          end if
23:       end for
24:    end for
25:    /* Move all triangles from B to A */
26:    emptyList(A)
27:    for all t in B do
28:       addElementToList(t, A)
29:    end for
30:    emptyList(B)
31: end while
32: return plane
End Function
```

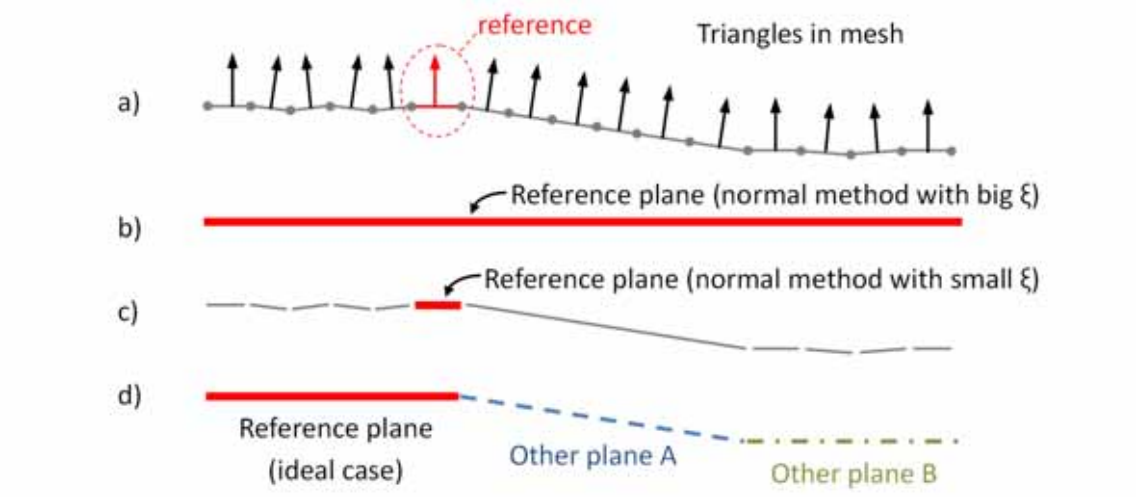


Figure 4.1: Plane detection using normal comparison

Here, a triangle belongs to the plane if the coordinates of its three nodes satisfy the equation $ax + by + cz + d$ according to a given threshold.

Similarly to the previous method, a triangle t is chosen as reference from the initial set S and its unit normal n_{ref} is computed. The plane equation parameters a, b, c correspond to respectively the three components of the unit normal vector $\bar{n}_{ref}$. Then the parameter d is obtained by inserting in the plane equation the coordinates of one of the three nodes of the reference triangle. The neighbour triangles are used to compute the value of $ax + by + cz + d$. If the value is lower than an imposed threshold ξ the concerned triangle is considered on the same plane as the reference triangle. For all the triangles inserted in the plane set, their neighbour triangles are considered. Algorithm 2 provides the pseudo-code description of this method.

The reference plane computed on a single triangle is not an optimal choice in the case of a noisy mesh such as scanned data. Figure 4.2 shows a comparison between using one and several reference triangles on perfect and noisy meshes. The three columns show separately the triangle(s) used for candidate plane computation, the computed planes and the triangles finally gathered from input triangles going through the reference plane. In case of perfect mesh one triangle is considered as reference triangle (fig.4.2.a) and the corresponding plane is computed (fig.4.2.b). With a certain tolerance all triangles are clustered (4.2.c). In case

Algorithm 2 Plane identification using plane equation computed on one triangle

Function planeEquationMethod (S As **List**, ξ As **Real**) As **Plane**

1: **Variable** $refT$, t, tt As **Triangle**
2: **Variable** $n1(3)$, $n2(3)$, $n3(3)$ As **Real**
3: **Variable** A, B As **List**
4: **Variable** a,b,c,d As **Real**
5: **Variable** *plane* As **Plane**
6: $refT \leftarrow S^0$
7: removeElementFromList($refT$, S)
8: addElementToList($refT$, $plane.Faces$)
9: addElementToList($refT$, A)
10: $plane.N \leftarrow$ normalOfTriangle($refT$)
11: planeEquationFromTriangle($refT$, a, b, c, d)
12: **while** $A \neq \Phi$ **do**
13: /* *Check the adjacency of triangles in A* */
14: **for all** t in A **do**
15: **for all** tt in S **do**
16: **if** isAdjacent(tt, t) **then**
17: $n \leftarrow$ normalOfTriangle(tt)
18: **if** $|a * n1(0) + b * n1(1) + c * n1(2) + d| \leq \xi$ **And**
$|a * n2(0) + b * n2(1) + c * n2(2) + d| \leq \xi$ **And**
$|a * n3(0) + b * n3(1) + c * n3(2) + d| \leq \xi$ **then**
19: addElementToList(tt, $plane.Faces$)
20: addElementToList(tt, B)
21: removeElementFromList(tt, S)
22: **end if**
23: **end if**
24: **end for**
25: **end for**
26: /* *Move all triangles from B to A* */
27: emptyList(A)
28: **for all** t in B **do**
29: addElementToList(t, A)
30: **end for**
31: emptyList(B)
32: **end while**
33: **return** *plane*

End Function

of noisy mesh a reference triangle is chosen (fig.4.2.d) and its plane is computed (fig.4.2.e). The triangle clustered maybe nothing except for the reference triangle. If the reference triangle A is chosen the plane is plane A. If the reference triangle B is chosen the plane B is computed.

It is easy to see that the plane represented by the set of triangles approximately does not pass by any triangle. Therefore, an intermediate plane fitting better two these triangles should be computed based on several reference triangles. Figure 4.2.g shows several reference triangles considered and the best fitting plane is computed (fig.4.2.h). With this best fitting plane more triangles could be clustered (fig.4.2.i).

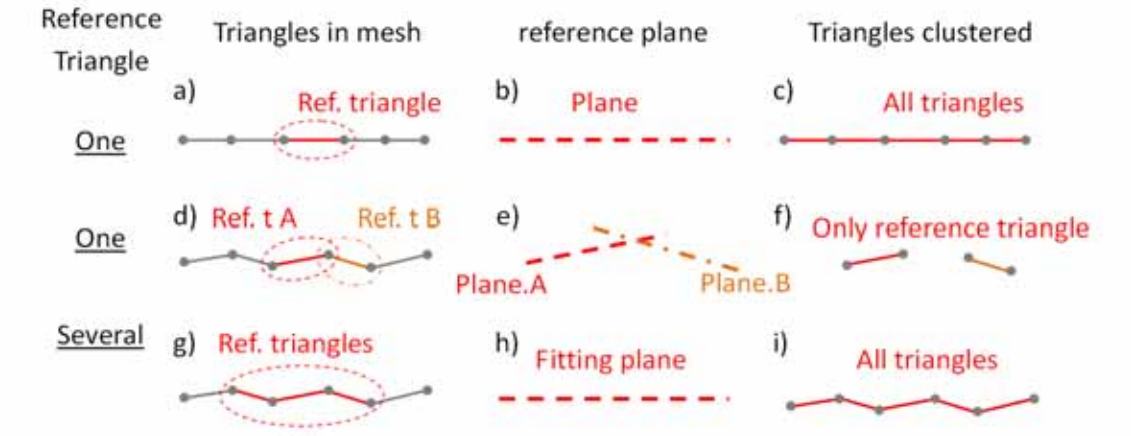


Figure 4.2: Plane equation computation based on one or several reference triangles

4.1.2.3 The best fitting plane with a set of triangles (adopted solution)

This method consists in taking a set of reference triangles and in computing the best fitting plane. The other faces are also gathered by checking the distance of the three nodes from the fitting plane according to a given tolerance.

The **first step** consists in determining the reference triangles. They are computed from a first reference triangle randomly chosen. The other triangles are n^{th} neighbours of the first reference triangle. The differences between the n^{th} neighbours faces' normals and the reference one should be null at the given tolerance ξ_1. For example, a triangle F is chosen as the first reference triangle (fig.4.3.a) and

its 1^{st} neighbours are the ones labeled by either "v" or "x". Only the ones labeled by "v" have a tolerable normals' difference. Therefore the reference triangles of 1^{st} neighbourhood are the blue ones in figure 4.3.b.

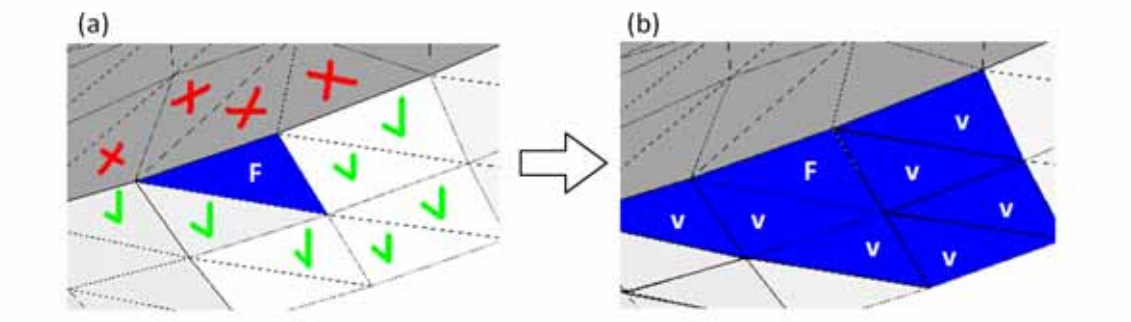


Figure 4.3: Selection of the reference faces with tolerable normal variation

The **second step** computes the plane equation that best fits the chosen reference triangles. To this aim, the weighted covariance matrix Cov_v of all the nodes associated with the reference faces is firstly computed. The normal $\vec{n}$ of the fitting plane is given by the eigenvector corresponding to the minimum eigenvalue of the matrix Cov_v. If the deviation from the fitting plane of the reference triangles is tolerable the fitting plane is accepted. The fitting plane computation is here detailed:

Let v_i be the vector (x_i, y_i, z_i) containing the coordinates of the node i, and its restricted Voronoi area $a(v_i)$ computed as:

$$a(v_i) = \sum_{j=1,\cdots,k} area(f_j)/k \tag{4.1}$$

where f_j are the triangles associated to v_i

Let $\bar{v}$ be the average coordinates for all the reference nodes computed by using the restricted Voronoi area as weight:

$$\bar{v} = \frac{\sum_i a(v_i) \times v_i}{\sum_i a(v_i)} \tag{4.2}$$

The covariance matrix Cov_v is computed in the following way:

$$Cov_v = \sum_i a(v_i)(v_i - \bar{v})(v_i - \bar{v})^T \tag{4.3}$$

When the unit normal n is computed (unit eigenvector corresponding to the minimum eigenvalue), a fitting error (e_f) is defined in order to verify whether the set of reference faces as well as the computed plane are acceptable. Of course, if there is only one face of reference, the fitting error is equal to zero and the computed plane is acceptable.

$$e_f = \sum_i a(v_i)(n(v_i - \bar{v}))^2 \tag{4.4}$$

If the fitting error is lower than a given threshold ξ_2 the best fitting plane is accepted and its equation is computed. The plane's parameter d is computed by solving the following equation.

$$n \cdot \bar{v} + d = 0 \tag{4.5}$$

The reference region is grown by seeking the neighbourhood triangles. The neighbour triangles are considered lying on the fitting plane if their three nodes v_i satisfy the ideal plane equation with a region-growth error (e_r) smaller than a given threshold ξ_3:

$$e_r = \sum_{i=1}^{3} (n \cdot (v_i) + d) \tag{4.6}$$

The algorithm for identifying a plane from a set of triangles by using the best fitting plane is detailed here in two parts (algos.3 and 4). The input list S contains a set of triangles on which planes have to be identified with the three thresholds that are given as input $\xi(3)$ in format of array .

4.1.3 Techniques to identify spherical areas

The technique adopted to identify a sphere areas on meshes containing triangular elements consists in determining the centre and radius of a sphere that best fits best a set of 3D points according to the least-squares criterion. The sphere

Algorithm 3 Plane identification using fitting plane computed on triangles PART-I

Function fittingPlaneMethod (S As **List**, $\xi(3)$ As **Real**) As **Plane**

Variable t, tt As **Triangle**
Variable $n1(3)$, $n2(3)$, $n3(3)$ As **Real**
Variable FT, A, B As **List**
Variable a,b,c,d As **Real**
Variable *plane* As **Plane**

/ Define reference triangles as well as the fitting plane equation */*

while $S \neq \Phi$ **do**
  $t \leftarrow S^0$
  removeElementFromList(t, S)
  addElementToList(t, FT)
  for all tt adjacentTo t **do**
    if isInList(tt, S) **then**
      if angleBetweenVectors(normalTriangle(t), normalTriangle(tt))$\leq \xi(0)$ **then**
        removeElementFromList(tt, S)
        addElementToList(tt, FT)
      end if
    end if
  end for

  fittingPlaneEquationFromTriangles(FT, a, b, c, d)
  if planeFittingError(FT, a, b, c, d)$\leq \xi(1)$ **then**
    exit while
  else
    emptyList(FT)
  end if
end while

if $FT = \Phi$ **then**
  return 0
end if

Algorithm 4 Plane identification using fitting plane computed on triangles PART-II

```
1:
2: /* plane region expansion */
3:
4: for all t in FT do
5:    addElementToList(t, plane.Faces)
6:    addElementToList(t, A)
7:    plane.N ← computeNormalOfPlane(a, b, c, d)
8:    plane.P ← computePointOnPlane(a, b, c, d)
9: end for
10:
11: while A ≠ Φ do
12:    for all t in A do
13:       for all tt adjacentTo t do
14:          if isInList(tt, S) then
15:             getThreeCordinatesFromTriangle(t, n1, n2, n3)
16:             if planeGrowingError(plane, n1, n2, n3)≤ ξ(2) then
17:                addElementToList(tt, plane.Faces)
18:                addElementToList(tt, B)
19:                removeElementFromList(tt, S)
20:             end if
21:          end if
22:       end for
23:    end for
24:
25:    emptyList(A)
26:    for all t in B do
27:       addElementToList(t, A)
28:    end for
29:    emptyList(B)
30:
31: end while
32: return plane
End Function
```

parameters' identification is performed on a set of triangles S chosen as reference, then the sphere coverage is computed by clustering the recursively surrounding triangles having the nodes satisfying the computed sphere equation.

The **first step** consists in the selection of the set S of reference triangles. Similar to the plane identification, a random triangle and its first neighbour triangles are chosen as reference triangles. Different from the case of plane identification, all the neighbour triangles are here considered. Figure 4.4 presents some example of reference triangle selection. The first reference triangular face (R^0) is chosen randomly from the initial triangle set. Then the neighbour triangles (R^1) associated through one of the three nodes of (R^0) are gathered into the reference triangle set.

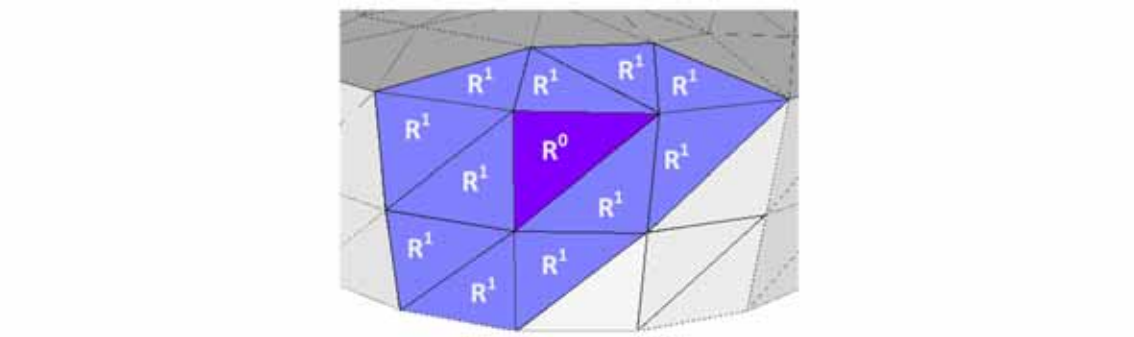


Figure 4.4: Selection of reference faces from one face (R^0) and its first neighbours (R^1)

The **second step** consists in determining the implicit equation 4.7 of the best fitting sphere to the reference triangles chosen.

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 - r^2 = 0 \tag{4.7}$$

The sphere is centered at point $c(c_x, c_y, c_z)$ and has a radius r. For computing the parameters of the fitting sphere, the vertex set V associated with the reference triangles in S are used within the least-square method. The squared Euclidean distance of a point $v_i(x_i, y_i, z_i)$ from sphere is :

$$d^2(v_i, S) = (\sqrt{(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2} - r)^2 \tag{4.8}$$

For all reference triangles, the best fitting sphere is computed to the associated points $(v_1 \cdots v_n)$ so that the sum of all the squared distances is minimised.

$$\min(\sum_{i=1}^{n} d^2(v_i, S)) \tag{4.9}$$

The above problem could lead to a resolution of a linear equation system using for example the Gauss-Newton method. The last might be a time consuming computation therefore the problem is simplified to find a good fitting using the concept of "algebraic distance". To do this the implicit equation 4.7 of the sphere can be written as:

$$x^2 + y^2 + z^2 + c_x^2 + c_y^2 + c_z^2 - 2c_x x - 2c_y y - 2c_z z - r^2 = 0 \tag{4.10}$$

which, in vector form, is equivalent to:

$$\begin{bmatrix} 2x & 2y & 2z & 1 \end{bmatrix} \cdot \begin{bmatrix} c_x \\ c_y \\ c_z \\ r^2 - (c_x^2 + c_y^2 + c_z^2) \end{bmatrix} = x^2 + y^2 + z^2 \tag{4.11}$$

The four unknowns c_x, c_y, c_z and r have been isolated in a single vector. Substituting x, y and z with the coordinates of the points in V, v_i $(x_i,\ y_i,\ z_i)$, we obtain the following over-determined linear system:

$$\underbrace{\begin{bmatrix} 2x_1 & 2y_1 & 2z_1 & 1 \\ 2x_2 & 2y_2 & 2z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 2x_1 & 2y_1 & 2z_1 & 1 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} c_x \\ c_y \\ c_z \\ r^2 - (c_x^2 + c_y^2 + c_z^2) \end{bmatrix}}_{w} = \underbrace{\begin{bmatrix} x_1^2 + y_1^2 + z_1^2 \\ x_2^2 + y_2^2 + z_2^2 \\ \vdots \\ x_n^2 + y_n^2 + z_n^2 \end{bmatrix}}_{b} \tag{4.12}$$

Therefore the unknowns could be solved directly using the following equations:

$$w = \begin{bmatrix} c_x \\ c_y \\ c_z \\ r^2 - (c_x^2 + c_y^2 + c_z^2) \end{bmatrix} = (A^T \cdot A)^{-1} \cdot A^T \cdot b \tag{4.13}$$

The fitting error (e_f) of the evaluated sphere with the reference triangle vertices v_i is computed. If the fitting error is lower than a threshold ξ_1 the fitting sphere is kept. The fitting error could be computed by the following formula:

$$e_f = \sum_i a(v_i) \cdot (\|v_i - c_i\|_2 - r)^2 \tag{4.14}$$

The **last** step concerns the sphere region growing which is performed by clustering the adjacent triangles that are approximately fitting the computed sphere. The criterion for clustering surrounding triangles into the sphere surface is the region-growth error (e_r) computed in a similar way than the equation 4.14. The nodes v_i used in the equation are then the three nodes associated with the triangle to test. If the region growing error is lower than a threshold ξ_2 the triangle is considered as being on the sphere.

$$e_r = \sum_{i=1}^{3} a(v_i) \cdot (\|v_i - c_i\|_2 - r)^2 \tag{4.15}$$

The algorithm for identifying a sphere from a set of triangles by using the best fitting sphere is detailed in two parts (algos.5 and 6). The input list S contains a set of triangles on which spheres have to be identified with the two thresholds that are given as input $\xi(2)$ in format of array .

4.1.4 Technique to identify cylindrical areas

Similarly to the previous shape fitting method, a set of triangles are chosen at first to compute the best fitting cylinder surface. For computing such a surface, three parameters have to be evaluated: a unit vector n parallel to the cylin-

Algorithm 5 Sphere identification using fitting sphere computed on triangles PART-I

Function fittingSphereMethod (S As **List**, $\xi(2)$ As **Real**) As **Sphere**

```
1: Variable t, tt As Triangle
2: Variable n1(3), n2(3), n3(3) As Real
3: Variable FT, A, B As List
4: Variable sphere As Sphere
5:
6: /* Define reference triangles as well as the fitting sphere equation */
7:
8: while S ≠ Φ do
9:    t ← S^0
10:   removeElementFromList(t, S)
11:   addElementToList(t, FT)
12:   for all tt adjacentTo t do
13:      if isInList(tt, S) then
14:         removeElementFromList(tt, S)
15:         addElementToList(tt, FT)
16:      end if
17:   end for
18:
19:   fittingSphereEquationFromTriangles(FT, sphere.P, sphere.R)
20:   if sphereFittingError(FT, sphere.P, sphere.R)≤ ξ(0) then
21:      exit while
22:   else
23:      emptyList(FT)
24:   end if
25: end while
26:
27: if FT = Φ then
28:   return 0
29: end if
```

Algorithm 6 Sphere identification using fitting sphere computed on triangles PART-II

```
1:
2: /* sphere region expansion */
3:
4: for all t in FT do
5:    addElementToList(t, sphere.Faces)
6:    addElementToList(t, A)
7: end for
8:
9: while A ≠ Φ do
10:   for all t in A do
11:      for all tt adjacentTo t do
12:         if isInList(ttt, S) then
13:            getThreeCordinatesFromTriangle(t, n1, n2, n3)
14:            if sphereGrowingError(sphere, n1, n2, n3)≤ ξ(1) then
15:               addElementToList(tt, sphere.Faces)
16:               addElementToList(tt, B)
17:               removeElementFromList(tt, S)
18:            end if
19:         end if
20:      end for
21:   end for
22:
23:   emptyList(A)
24:   for all t in B do
25:      addElementToList(t, A)
26:   end for
27:   emptyList(B)
28:
29: end while
30: return sphere
End Function
```

der axis, a centre point c on the cylinder axis and the radius r of the cylinder. Once the cylinder is determined the region is grown by clustering the appropriate surrounding triangles.

The **first step** is to define a set of reference faces for computing the best fitting cylinder surface. It is done exactly in the same way as for the sphere identification (fig.4.4).

The **second step** is the computation of the cylinder parameters. The vector n parallel to the cylinder axis is defined by finding the direction on which the variation of reference faces normal is minimal and at the same time maximal in all the orthogonal directions. In static point of view this direction vector n is the eigenvector corresponding to the maximum eigenvalue of a covariance matrix Cov_v. that is represented as following equation:

$$Cov_v = \sum_i \|e_i\| \cdot \beta(e_i) \cdot \bar{e}_i \times \bar{e}_i^T \tag{4.16}$$

where e_i is one of edges of the reference triangles, $|e_i|$ is the edge length, $\bar{e}_i$ is its unit vector parallel to the edge and $\beta(e_i)$ is the angle between the normals of the two triangles sharing the edge e_i. Once having determined the cylinder axis vector, the estimation of a point c on the axis and the radius r is equivalent to compute the centre and radius of the circle resulting from the intersection between the cylinder and the plane orthogonal to the axis. For computing the circle, all nodes related to the reference triangles are projected onto the plane orthogonal to the axis, and then a 2D circle that best fits these projected points is computed. The 2D fitting circle could be determined with the method for fitting sphere described in the previous section by reducing the dimensionality to 2.

The orthogonal plane normal is equivalent to the cylinder axis vector n and the passing point c_m could be computed as the barycentre of the projected nodes. An orthonormal basis $B = \{n, e_x, e_y\}$ is created for the orthogonal plane and e_x, e_y could be the remaining two eigenvectors of the covariance matrix Cov_v that are different from the one equivalent to n.

The 3D coordinates v_i of all nodes related to the reference triangles are transformed into 2D: $B(v_i) = \{(v_i - c_m)e_x, (v_i - c_m)e_y\}$, on the basis B. The 2D

coordinates of the centre $O = (o_x, o_y)$ of the 2D circle can then be transformed into the 3D coordinates for the cylinder centre $c = c_b + e_x o_x + e_y o_y$.

When the cylinder parameters (the axis vector n, the center c and the radius r) are computed, a fitting error (e_f) related to the reference triangle nodes v_i ($i = 1, .., n$) is computed by the following equation:

$$e_f = \sum_{i=1}^{n} a(v_i)(\|(v_i - c) \times n\|_2 - r)^2 \tag{4.17}$$

If the current fitting error is inferior to a given threshold the best fitting cylinder is accepted.

The **last step** is to grow up the cylinder region by clustering the surrounding triangles (three vertices v_1 , v_2 , v_3) under condition of region-growth error (e_r), which is computed by the following formula, lower than the given threshold:

$$e_r = \sum_{i=1}^{3} a(v_i)(\|(v_i - c) \times n\|_2 - r)^2 \tag{4.18}$$

The detailed algorithm for cylinder surface identification is written in two parts (algo.7 and 8).

4.1.5 Freeform

The techniques to identify the three basic shapes of plane, sphere and cylinder have been presented in the previous sections. Even if these three types do not cover all the possible shapes a product, our analyses based of industrial EDF model used in maintenance studies shows that these three shapes are the most recurrent. To provide a complete classification of the product "skin", a part of surface on which none of the three basic shapes is recognised, is classified into freeform shape category.

4.1.6 Overall algorithm and examples

In this subsection, the use of the previously introduced elementary shape recognition techniques is presented. These techniques can be applied either for segmen-

Algorithm 7 Cylinder surface identification using fitting cylinder computed on triangles PART-I

```
Function fittingCylinderMethod (S As List, ξ(2) As Real) As Cylinder
 1: Variable t, tt As Triangle
 2: Variable n1(3), n2(3), n3(3) As Real
 3: Variable FT, A, B As List
 4: Variable cylinder As Cylinder
 5:
 6: /* Define reference triangles as well as the fitting cylinder equation */
 7:
 8: while S ≠ Φ do
 9:   t ← S^0
10:   removeElementFromList(t, S)
11:   addElementToList(t, FT)
12:   for all tt adjacentTo t do
13:     if isInList(tt, S) then
14:       removeElementFromList(tt, S)
15:       addElementToList(tt, FT)
16:     end if
17:   end for
18:
19:   fittingCylinderEquationFromTriangles(FT, cylinder.P, cylinder.N,
      cylinder.R)
20:   if cylinderFittingError(FT, cylinder.P, cylinder.N, cylinder.R)≤ ξ(0)
      then
21:     exit while
22:   else
23:     emptyList(FT)
24:   end if
25: end while
26:
27: if FT = Φ then
28:   return 0
29: end if
```

Algorithm 8 Cylinder surface identification using fitting cylinder computed on triangles PART-II

```
1:
2: /* cylinder region expansion */
3:
4: for all t in FT do
5:    addElementToList(t, cylinder.Faces)
6:    addElementToList(t, A)
7: end for
8:
9: while A ≠ Φ do
10:   for all t in A do
11:      for all tt adjacentTo t do
12:         if isInList(ttt, S) then
13:            getThreeCordinatesFromTriangle(t, n1, n2, n3)
14:            if cylinderGrowingError(cylinder, n1, n2, n3)≤ ξ(1) then
15:               addElementToList(tt, cylinder.Faces)
16:               addElementToList(tt, B)
17:               removeElementFromList(tt, S)
18:            end if
19:         end if
20:      end for
21:   end for
22:
23:   emptyList(A)
24:   for all t in B do
25:      addElementToList(t, A)
26:   end for
27:   emptyList(B)
28:
29: end while
30: return cylinder
End Function
```

tation of the whole mesh into surfaces, or for understanding whether the elements composing groups are presenting specific shapes. In both cases, the type of surface shape is considered to be an important characteristic to be preserved during the application of the CAD-less mesh modelling operators.

By giving a triangle mesh, all the triangles are used to identify sequentially planes, spheres and cylinders. The word "sequentially" indicates that all triangles recognised as "belonging" to the identified planes will not be used to identify the two other types of shapes (sphere and cylinder). Conversely, the triangles on which no plane is detected are used to identify spheres, finally only the triangles on which no planes and spheres are recognised are used to recognise the cylinder shape.

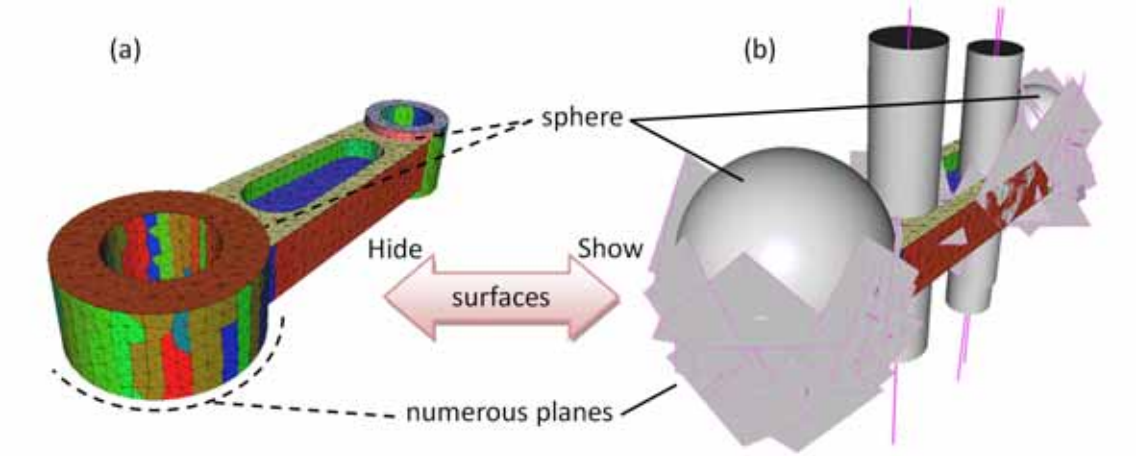


Figure 4.5: Example of shape detection without triangle minimum number control

Since a local part of a sphere or a cylinder could be very similar to a plane and a local part of a cylinder could be very similar to a sphere as well as vice-versa, therefore a notion of "minimal number of triangles" is introduced to avoid the recognition of local non meaningful partial shapes. The shape detection without minimum triangles controlling on a connecting rod mesh model is illustrated in figure 4.5. The picture (a) shows the identified groups of triangles, each group presents a different shape whose surface is shown in the picture (b). In this mesh model four cylinders are present but two of them are recognised as numerous wrong planes and spheres. Each wrongly detected plane or sphere is represented

by a group of triangles of small cardinality in comparison with the number of triangles lying on the two cylinders (fig.4.5.d).

The global algorithm for shape detection is detailed in algorithm 9. The input parameters consist of a mesh, three threshould arrays and three minimal number of triangles respectively for three shape identification. Several tests have been carried out to identify automatisms for automatic computation but a general sound solution seems not sufficiently reliable for any possible model. Thus, the algorithm is implemented so that the different shapes are identified under the user supervision. This shape information belongs to what we refer to as shape semantics. As shown in the following algorithm certain instances of different shape data structures (see section 4.1.1 on page 86) are created at the end of the process. They are used to enrich the FE mesh model over the structure and semantics layer (fig.3.1). Thus, each recognised surface (including its parameters) is stored at the semantic layer, and is supported at the structural level by a group collecting the corresponding mesh elements.

Some examples of shape recognition are shown in figure 4.6. Figure 4.6.a presents a cube-like model on which the initial sharp edges are rounded. On this model there are six planar facets twelve cylinder and eight spherical parts on the intersection zone between the cylindrical parts. Figure 4.6.b shows a model presenting two cylinders sharing the same axis and having different radius. A half of sphere on the top and three planes are also recognised in this model. The connecting rod is shown in figure 4.6.c the different colors correspond to the various surfaces correctly detected which are shown in figure 4.6.d. On the three models no freeform shape parts are present since all the triangles belong to areas defining one of the three basic shapes mentioned above.

The presented shape recongnition techniques could be adapted to other kinds of surface meshes such as quadrangle. It is because the technique of shape fitting is based on the nodes therefore the presented technique does not depend on the mesh elements.

Algorithm 9 Planes, spheres and cylinders identification from a given mesh

Function shapeRecognitionOnMesh (M As **Mesh**, $\xi 1(3)$ As **Real**, $\xi 2(2)$ As **Real**, $\xi 3(2)$ As **Real**, $m(3)$ As **Integer**) As **List**

```
1:
2: Variable P, S, C As List
3: Variable Shapes As List
4: Variable plane As Plane
5: Variable sphere As Sphere
6: Variable cylinder As Cylinder
7: for all triangles t in mesh M do
8:    addElementToList(t, P)
9:    addElementToList(t, S)
10:   addElementToList(t, C)
11: end for
12:                  /* planes identification */
13: while P ≠ Φ do
14:   plane ← fittingPlaneMethod(P, ξ1)
15:   if cardinalityOfList(plane.Faces)≥ m(0) then
16:      addElementToList(plane, Shapes)
17:      for all t in plane.Faces do
18:         removeElementFromList(t, S)
19:         removeElementFromList(t, C)
20:      end for
21:   end if
22: end while
23:                  /* sphere identification */
24: while S ≠ Φ do
25:   sphere ← fittingSphereMethod(S, ξ2)
26:   if cardinalityOfList(sphere.Faces)≥ m(1) then
27:      addElementToList(sphere, Shapes)
28:      for all t in sphere.Faces do
29:         removeElementFromList(t, C)
30:      end for
31:   end if
32: end while
33:                  /* cylinder identification */
34: while C ≠ Φ do
35:   cylinder ← fittingCylinderMethod(C, ξ3)
36:   if cardinalityOfList(cylinder.Faces)≥ m(2) then
37:      addElementToList(cylinder, Shapes)
38:   end if
39: end while
40: return Shapes
```

End Function

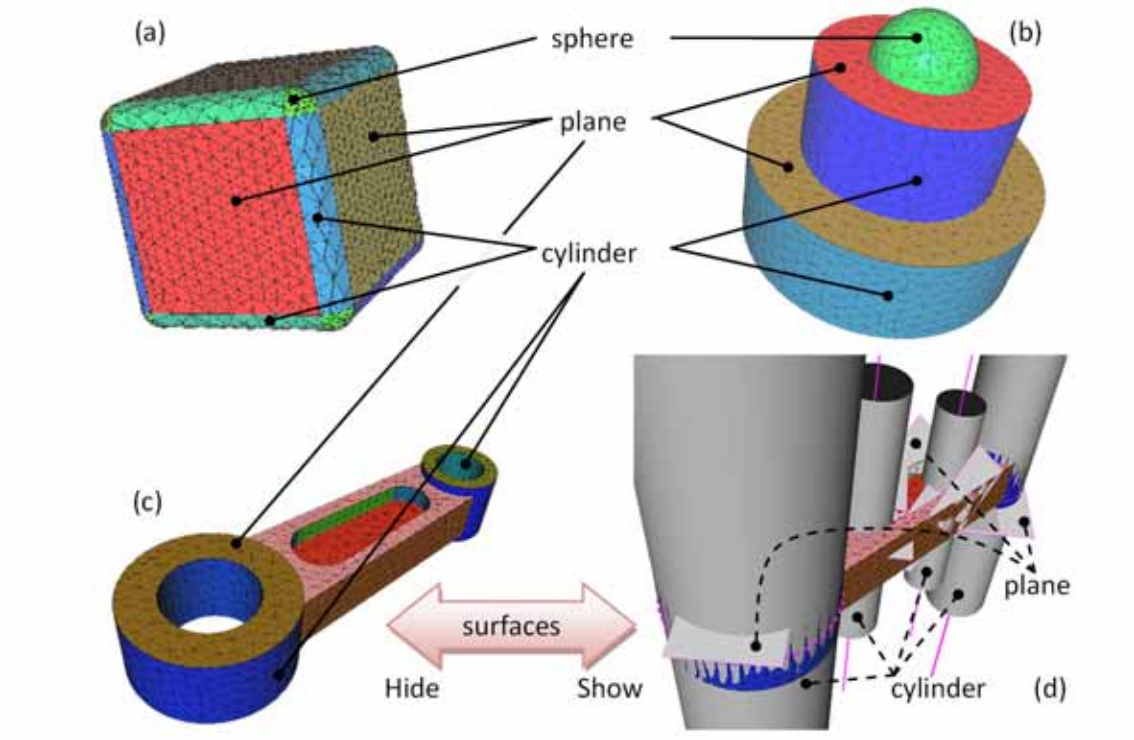


Figure 4.6: Basic shapes recognition examples

4.2 Sharp feature recognition

In this section the technique for sharp feature detection on mesh [87] is presented. A sharp feature in a solid consists of creases, corners, and darts [44]. A crease is a tangent line smooth curve along which the surface is C^0 but not C^1. A corner is a point where three or more creases meet. A dart is an interior point of a surface where a crease terminates. Although this list of sharp features is not exhaustive (for instance, we cannot model a cone or two coincident darts), it is sufficient for examples we have encountered. A simple example having different sharp features is shown in figure 4.7.

Regarding meshes, the sharp features are supported by a set of sharp edges and vertices. A sharp edge is defined as following [40]: an edge shared by two triangles whose normal vectors make a dihedral angle higher than a given threshold. Vertices that belong to a sharp edge are considered as sharp vertices, but an edge shared by two sharp vertices is not necessarily a sharp edge. For differ-

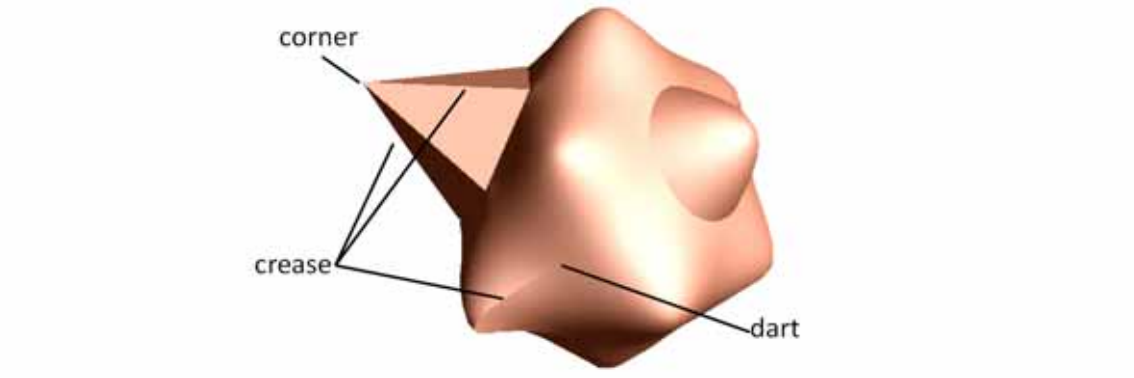


Figure 4.7: Sharp feature examples [44]

ent angle thresholds, different sharp edges will be detected. In this manuscript, the automated choice of the threshold is not talked about and this parameter is user-specified. The proposed CAD-less mesh modelling operator includes this preliminary sharp feature detection step.

4.2.1 Sharpness weight on an edge

A concept about sharp edge is provided in the paper [47]. The classification operator assigns a weight w to every edge e in a mesh and the weight $w(e)$ is proportional to the importance of the edge. Ideally, edges close to or on crease-type features should be assigned large weights whereas all remaining edges should get small weights. According to the author, the value of the weight is marked to distinguish sharp edges from the others. There are four ways to define the weight of edge:

1. Second order Difference (SOD)
   The easiest way to assign a weight to an edge shared by two triangles is to compute the dihedral angle between the normals of these two triangles. The formula for computing the dihedral angle is described by the equation:

$$w(e) = \cos\left(\frac{n_i}{\|n_i\|} \cdot \frac{n_j}{\|n_j\|}\right)^{-1} \tag{4.19}$$

   where $w(e)$ is the dihedral angle, n_i and n_j correspond to the normals of the two triangles sharing the edge e. Figure 4.8.a shows that two normals are

taken from the two triangles sharing an edge for computing the angle. This approach is well fitting to coarse or pre-optimised meshes. Whereas it leads to poor results on very smooth or noisy meshes, since a very small region around the concerned edge is taken into account for all computations.

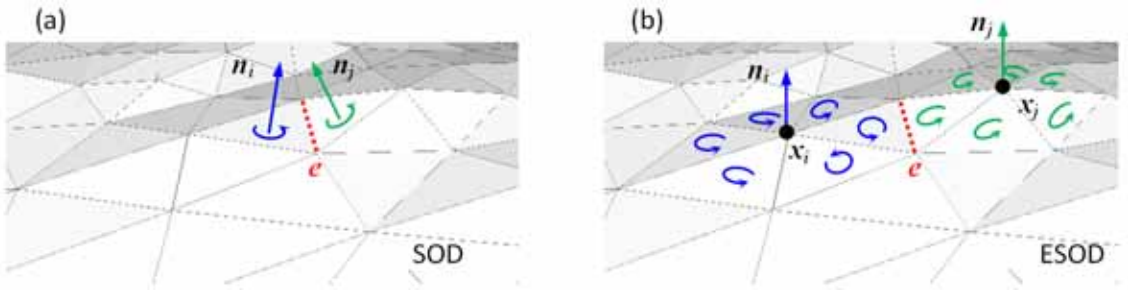


Figure 4.8: Sharp feature examples [44]

2. Extended Second Order Difference (ESOD)
   To overcome the above limit a simple extension of the previous operator (*SOD*) can be considered by taking the mean normal computed from the one-ring of the vertices opposite to the edge e to replace the normal of two neighbour triangles, as shown in figure 4.8.b. The two normals n_i and n_j are computed on the two vertices x_i and x_j opposite to the edge e by using their surrounding triangles normals. By the extension of the normal computation, the operator is improved by reducing the influence of noise. However comparing with the operator *SOD*, *ESOD* has lost the advantage in dealing with the coarse meshes.

3. Best Fit Polynomial (BFP)
   In the operator *ESOD*, only the triangles around the vertex are taken into account in the computation of the normal. In order to obtain more support from the circumjacent vertices, the operator *BFP* is proposed [47]. In this approach a projection plane being perpendicular to the considered edge is used as an assistant parameter. In addition, the mid-point of the edge is defined on the intersection between the parameter plane and the edge, as shown in figure 4.9. The points computed from the intersection of the plane with a set of circumjacent triangle edges are then interpolated with a

best fit polynomial $P(u)$ of degree n. Finally the curvature of the (planar) polynomial is evaluated at the edge position e (parametric coordinate $u = u_e$), as described by equation:

$$w(e) = P''(u_e) \tag{4.20}$$

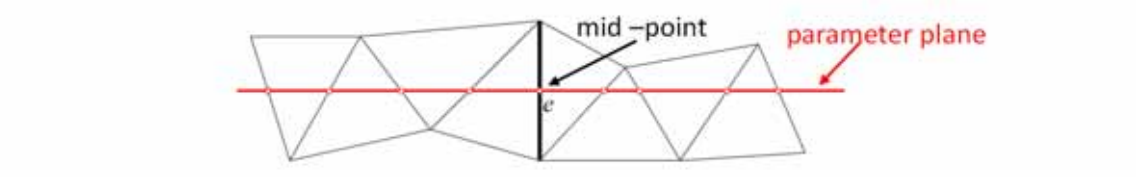


Figure 4.9: Top view of a mesh with the concerned edge e and the parameter plane used for the *BFP* method [47]

4. Angle Between Best Fit Polynomials (ABBFP)
   The *ABBFP* is an extension of the *BFP*. As in the *BFP* case, polynomials are fitted through the parameter plane of every edge. The *ABBFP* operator contains two polynomials: one for the vertices lying on one side of the edge, and one for the vertices lying on the other side, as shown in figure 4.10. The angle between the two curve tangents at the edge position is chosen to assign the weight of edge, as described by equation:

$$w(e) = \cos\left(\frac{(1, P'_\ell(e))}{\|(1, P'_\ell(e))\|} \cdot \frac{(1, P'_r(e))}{\|(1, P'_r(e))\|}\right)^{-1} \tag{4.21}$$

   The $P_\ell(e)$ is the left polynomial and whereas the $P_r(e)$ is right polynomial.

This section has shown four ways to compute sharpness for classifying edges. The approaches are based on the surface normal variation from one side of the edge to another side. Each approach corresponds to a different way to compute the normal on the two sides and the range is considered far or near to the edge. Once the sharpness is defined on each edge the relative sharp edge are defined. By giving a threshold the sharp edges could be clustered. The choice of the threshold could be human-aided determination or automated.

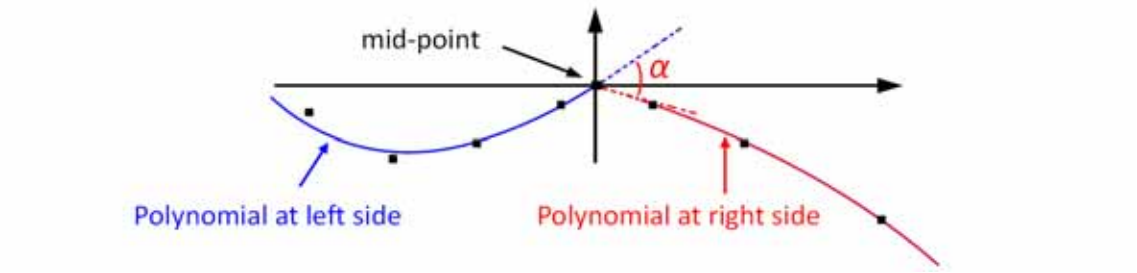


Figure 4.10: Angle between the two polynomials used for the ABBFP method (side view) [47]

4.2.2 Discrete curvature on a node

In the previous subsection, four approaches for computing the sharpness of an edge have been introduced. In this section the discrete curvature computed on a node is introduced. The edge is weighted for sharp edge detection by using the curvature of the two extreme nodes. As we know, a mesh represents the shape geometry in a discretised way, which leads to the curvature being discrete.

The discrete curvature at a node p could vary according to the length of the edges associated with the node p (fig.4.11). Therefore, invariant for the discrete absolute curvature at a node is applied. For the node p, the invariant of the discrete absolute curvature is the combination of the invariant of the discrete Gaussian curvature at the node p with the invariant of the discrete mean curvature at the node p [59].

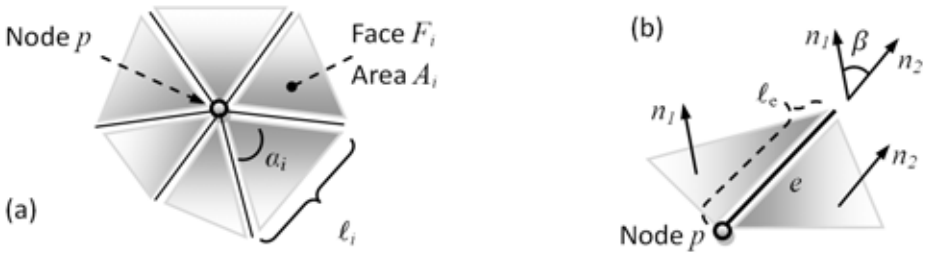


Figure 4.11: (a) Geometric parameters of a star-set associated to a node p; (b) edge length and dihedral angle β [59]

1. Discrete Gaussian curvature

   The discrete Gaussian curvature K_p of a triangulation at a node p (fig.4.11.a)

is computed by the ratio of the round angle deviation on the node p to the area of the surrounding triangles (called ***star-set***):

$$K_p = \frac{2\pi - \sum_i \alpha_i}{\frac{1}{3}\sum_i A_i} \tag{4.22}$$

where A_i is the area of the i^th face which shares the node p with other faces and α_i is the angle of this face.

As long as the triangles of the polyhedron are quite regular, this approximation will lead to a good approximation. That's to say that this approximation gives good result only to the nodes whose associated triangles have similar areas and are nearly equilateral. For producing a more robust approximation of the Gaussian curvature in most of the cases, and mainly when triangles have irregular areas and aspect ratios, a new formulation based on the so-called ***modulus*** of area of the star-set which is different from the last equation is introduced:

$$K_p = \frac{2\pi - \sum_i \alpha_i}{\frac{1}{2}\sum_i A_i - \frac{1}{8}\sum_i \cot(\alpha_i)\cdot \ell_i^2} \tag{4.23}$$

where A_i is the area of the i^th face which shares the node p with other faces, α_i is the angle of this face and ℓ_i is the length of the edge opposite to the node p (fig.4.11).

2. Discrete mean curvature

   The discrete mean curvature He is basically defined for an edge by the following equation.

$$H_e = \frac{1}{2}\cdot\beta\cdot\ell_e \tag{4.24}$$

   where ℓ_e is the length of the edge being considered and β is the angle between the normals of the adjacent faces which share the current edge (fig.

   In the presupposition that H_e is uniformly spread along the edge, the for-

mula of the discrete mean curvature at a node p can be expressed as equation 4.25. The value of H_{ep} around p incorporates the half sum of the value H_e of the edges around the node p and enables the definition of curvature quantities at the same location, i.e. at the node.

$$H_{ep} = \frac{1}{4} \cdot \sum_j \beta_j \cdot \ell_{ej} \tag{4.25}$$

When the approximation takes into account the non uniformity of the triangulation, the value of the discrete mean curvature at a node p can be evaluated by the extension of the equation 4.25. If the value H_{ep} is divided by the area of the star-set, the formula can express the discrete mean curvature H_e (eq.4.26). If the value H_{ep} is divided by the modulus of the same star-set attached to p, it gives the equation 4.27.

$$H_p = \frac{\frac{1}{4} \cdot \sum_j \beta_j \cdot \ell_{ej}}{\frac{1}{3} \sum_i A_i} \tag{4.26}$$

$$H_p = \frac{\frac{1}{4} \cdot \sum_j \beta_j \cdot \ell_{ej}}{\frac{1}{2} \sum_i A_i - \frac{1}{8} \sum_i \cot(\alpha_i) \cdot \ell_i^2} \tag{4.27}$$

Comparing equation 4.27 with equation 4.23, they are apparently sharing the same denominator. The experiments carried out in the paper [59] prove that the equation 4.27 tends to produce a better mean curvature approximation than equation 4.26, since the values provided by equation 4.27 are less sensitive to the aspect ratio of the star-set around a vertex. Therefore equation 4.27 is selected to compute the value of the discrete mean curvature at a vertex.

3. Discrete absolute curvature

   Generally speaking, the absolute curvature K_{abs} at a node is defined as the sum of the squared principal curvatures 4.27. A combination of the mean and the Gaussian curvatures can also evaluate the absolute curvature.

$$K_{abs} = k_1^2 + k_2^2 = 4H^2 - 2K \tag{4.28}$$

where k_1 and k_2 are two principal curvatures, H expresses the mean curvature while K express the Gaussian curvature.

4. Invariant criteria based on discrete curvature
   The discrete curvatures presented previously are very meaningful on a surface C^2. In case the node to be evaluated is lying on sharp lines, the value of the discrete curvatures is no longer meaningful. Figures 4.12.a, 4.12.b and 4.12.c, show for a same solid, three different tessellations around node p. The number and size of triangles associated with p are different even if the underlying surface is the same (corner of a box). With the previously presented discrete curvature computations the results on the three meshes are different. Therefore the curvature computation algorithm on a node p should be elaborated in order to be independent of the mesh connectivity and shape of the triangles around the node p. The criteria should rely as much as possible on the form of the underlying smooth surface described by the mesh.

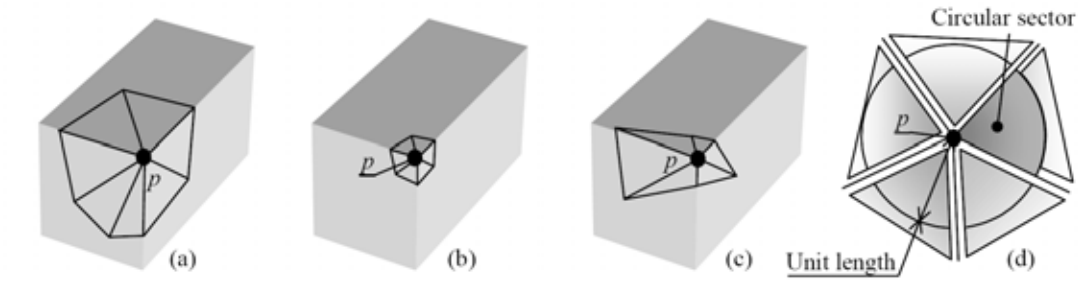


Figure 4.12: (a,b,c) the same geometry at node p is described by equivalent meshes, (d) circular sectors associated to the star-set of the node p [59]

The solution to stabilise the curvature computation is to replace the star-set triangles at the node p by a set of unit circular sector (fig.4.12.d). Thus in many previous formulas, the area of the star-set triangles will be replaced by the half sum of the sector angles. The area of a circular sector with a radius r and an angle α is:

$$A = \frac{1}{2} r^2 \cdot \alpha \tag{4.29}$$

After replacing the star-set of triangles at node p by a star-set of unit circular sectors ($r = 1$), the area of a sector associated with the node p can

be computed by the formulation, which is the approximation of the area of unit circular sector:

$$A = \frac{1}{2}\alpha \tag{4.30}$$

The length of the edge opposite to p can be approximated by the formula:

$$\ell_i = \alpha \cdot r = \alpha \tag{4.31}$$

The area of the start-set of a unit circular sector is therefore:

$$\sum_i A_i = \frac{1}{2}\sum_i \alpha_i \tag{4.32}$$

Using this transformation, the new criteria for the sharp lines detection are invariant for equivalent meshes and are expressed by using the unit length circular sector instead of the triangles. The different curvature equations presented previously are re-written in order to get invariant:

- Invariant for the discrete mean curvature (eq.4.24) along an edge:

$$I_{H_e} = \frac{1}{2}\beta \tag{4.33}$$

- Invariant for the discrete mean curvature at a node p:

$$I_{H_p} = \frac{3}{2} \cdot \frac{\sum_j \beta_j}{\sum_i \alpha_i} \tag{4.34}$$

  The formula is obtained from equation 4.26. Similarly, another expression of I_{H_p} can be obtained from equation

- Invariant for the discrete Gaussian curvature (eq.4.23) at a node p:

$$I_{K_p} = \frac{2\pi - \sum_i \alpha_i}{\frac{1}{4}\sum_i \alpha_j - \frac{1}{8}\sum_i \alpha_i^2 \cot(\alpha_i)} \tag{4.35}$$

- Invariant for the discrete absolute curvature (eq.4.28) at a node p:

$$I_{K_{absp}} = 4I_{H_p}^2 - 2I_{K_p} \tag{4.36}$$

4.2.3 Use of sharp feature detection in CAD-less mesh modelling operator

The invariant for discrete absolute curvature is adopted to compute the sharpness quantity on nodes. The nodes are considered as sharp if they have a higher value than a certain threshold given by the user. After the computation of the invariant of the discrete absolute curvature, the weight of an edge can be computed by the average value of invariant of the discrete absolute curvature of the two nodes at the end-points. Therefore, edges associating two sharp nodes can be tagged as sharp edges.

In figure 4.13, sharp feature detection is performed on several mesh examples. The sharp nodes as well sharp edges are highlighted in red color. These meshes are either triangle mesh or tetrahedral mesh. In case of a tetrahedral mesh, the sharp feature detection is only performed on the outer hull defined by a triangle mesh.

Concerning the proposed CAD-less mesh modelling operator, the sharp feature information is enriched on the FE mesh model over the structure and semantics layers in the early stage (fig.3.1). New groups of mesh nodes that have a curvature higher than the user-specified threshold are created. Nodes having equivalent curvature values are clustered in a same group. Therefore, each newly created group support a semantic information relative to the curvature value of the identified sharp features.

During geometry modification, these sharp features are either used as a constraint or as a guide for mesh modification operator:

- During the modification, if the position of a sharp node is not fixed, the constraint of keeping the sharpness could be applied. For example, when merging two intersecting meshes, not only the intersection zone is modified but also the surrounding. The surrounding is modified in order to smooth the transition from the intersection zone to the unchanged zone. If there are

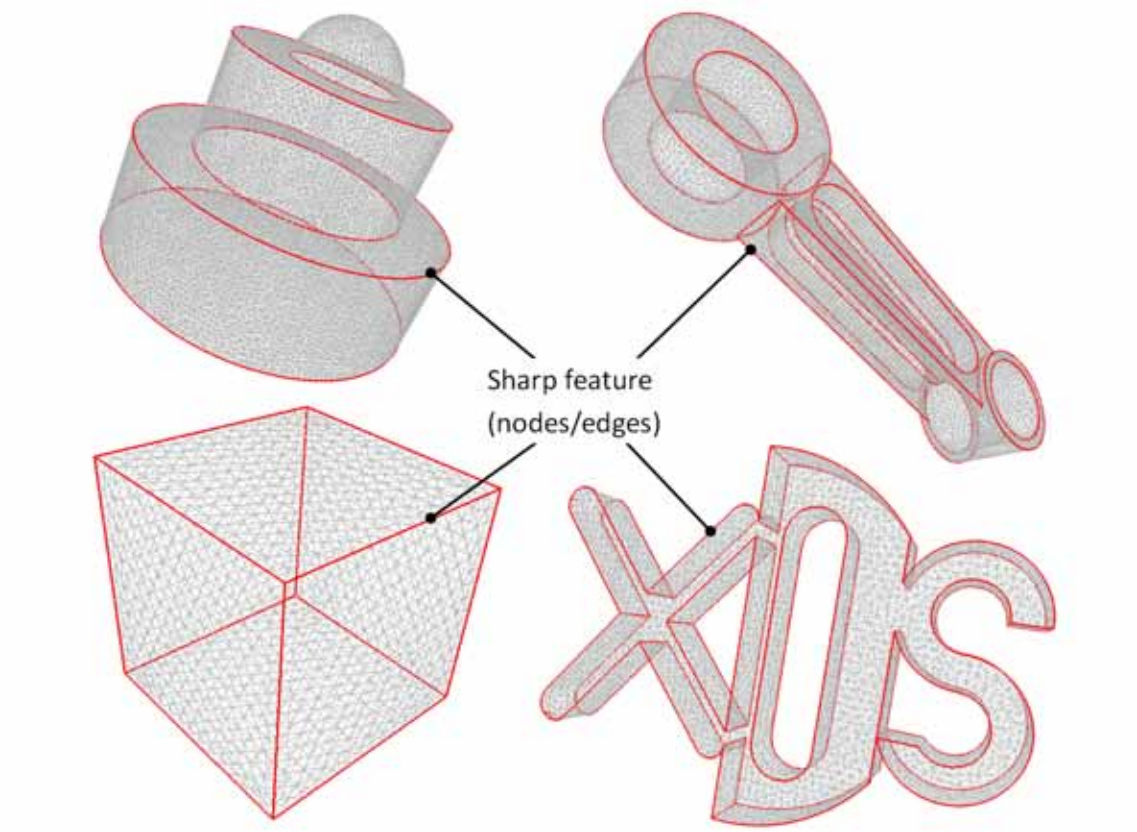


Figure 4.13: Examples of sharp feature detection within our tool

sharp nodes in the transition zone it is better that these nodes move along the sharp feature (creases) to relax the transition (section 6.2 on page 181).

- During the mesh modifications requiring a change of the sharp feature, the early prepared classification of nodes based on the curvature quantity can be used to directly find the sharp feature. For example, in case of the filleting operation on FE mesh, when the user chose an edge of the mesh to do the filleting, all other connected edges within the same curvature group can be also taken into account simultaneously (section 6.5 on page 238).

4.3 Mesh quality checking

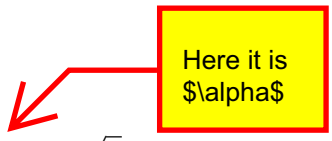
Since the quality of the FEA relies also on the quality of the used meshes, it is crucial to be able to control and maintain the mesh quality during the modification phases. Here, mainly two criteria are checked and therefore introduced in this section. The first one refers to the aspect ratio of the mesh elements (triangles/tetrahedra), and the second one deals with the self-intersection problem.

Of course, since the approach is modular, other criteria could be checked and integrated within the CAD-less mesh modelling operator.

4.3.1 Aspect ratio of elements

For triangle meshes, a degenerated triangle is characterised by exactly one or two small angles. To quantify the degeneracy without computing the three angles, we use an indicator introduced in [24] to check the quality of FE meshes and defined as follows:

$$Q = \alpha \frac{S}{hp} \tag{4.37}$$



where Q is the aspect ratio of a triangle with $alpha = 2\sqrt{3}$ a normalisation coefficient so that $Q = 1$ for an equilateral triangle, h is the longest edge length, S is the area of the triangle and p its half-perimeter. This quality factor belongs to the interval [0, 1]. The limit 0 corresponds to flat triangles. It is commonly accepted that a triangulation is a good one, with respect to the FE approximation, if the aspect ratio of the worst triangle is greater than 0.5. Similarly, for tetrahedral meshes, the definition of the quality of a tetrahedron is introduced in [12]:

$$Q = \alpha \frac{\rho}{h_{max}} \tag{4.38}$$

where h_{max} is the length of the longest edge in the tetrahedron, ρ is the radius of the inscribed sphere and $\alpha = 2\sqrt{6}$ is a normalisation factor so that the quality of an equilateral tetrahedron is 1. This quality function varies in the interval [0, 1] (a well-shaped tetrahedron has a quality close to 1, while an ill-shaped element has a quality close to 0).

4.3.2 Self-intersection

The second mesh quality criterion checks whether there are self-intersections. Self-intersections occur when a part of a mesh collides with another part of itself, i.e. two elements of the same mesh intersect each other. A self-intersection destroys the integrity of the mesh and makes the mesh unusable for FEA applications. In triangle or tetrahedral meshes the check of the intersection between edges and triangles is necessary.

4.4 Topological operations

In the developed CAD-less mesh modelling operator, a topological operation may concern an addition or a removal of mesh entities of different dimension, and consequently it consists also in changing the connections between the mesh elements.

The triangulation and tetrahedralisation add mesh elements of dimension equal to mesh. The mesh refinement adds mesh elements of lower dimension. The mesh deletion consists in removing the mesh entities of dimension equal to mesh whereas the mesh simplification eliminates mesh elements of lower dimension. The mesh entities duplication is used to disconnect two sub-meshes of the mesh. The swap of the mesh elements changes the connection between mesh entities for improving the mesh quality. All these operations are further detailed hereafter since they concur to the definition of the CAD-less mesh modelling operators.

4.4.1 Triangulation in edge loop

The triangulation algorithm aims at creating new triangles by giving a closed edge loop. There are many approach already proposed and presented in the chapter 2 (p.48). This is useful in filling holes in meshes for repairing models [89] and for re-meshing intersection zone to merge disconnected mesh parts [76]. The first work uses the triangulation to fill holes in meshes by preserving the curvature given in the surrounding mesh. The second work is using triangulation for re-meshing the intersection zone while the two meshes are intersecting. They both use a triangulation method based on the one proposed in [60].

For prototyping our CAD-less mesh modelling operator, a modified version of the second method is proposed. The re-developed triangulation technique takes as input a set of closed edge loop and a weight for each node of the possible triangulation. The whole process will weight from the smallest polygon (with two nodes) until the biggest polygon (from the bottom to top). Each polygon's weight is computed from the minimum weight given by the different possible combination of the sub-polygons and the combination is registered. Then the triangulation starts from the global polygon by defining the triangle on one edge so that the sum of the weights for the sub-polygon is minimal. Each sub-polygon will continue to find its sub-polygon with minimal weight sum.

Figure 4.14 shows an example of the hierarchical triangulation on an initial polygon (fig.4.14.a):

- We start from computing the weight of all possible polygons with 3 nodes (triangles). In the existing technique, the weight is the area of the triangle. At the bottom level figures 4.14.q - 4.14.t show certain polygons with 3 nodes.

- The polygon with 4 nodes at level "*N-3*" will adopt a triangulation on which the weight sum of the sub-polygons is minimal. For example figures 4.14.k and 4.14.l present two ways of triangulation on the same polygon with 4 nodes. In the triangulation (fig.4.14.k) the weights of the two sub-polygons with 3 nodes ($w23$ and $w24$) are already computed in the lower level (fig.4.14.q and fig.4.14.r). The weight of this polygon will be the smaller weight between the ones given by the two triangulations and the corresponding triangulation is registered in the polygon.

- The polygons on the level "*N-2*" will be the triangulation of the smallest sum of the weights on the sub-polygons. The sub-polygons could be from the level "*N-3*" or "*N-4*". For example the weight of the triangulated polygon in figure 4.14.h is computed by sum of the weights $w16$ and $w17$. The first weight is computed from the minimal weight between the two polygons (fig.4.14.k and fig.4.14.l) on the level "*N-3*" and the second weight is computed from the polygon (fig.4.14.u) on the level "*N-4*".

- We continue upstairs, at level "*N-1*". The polygons' weight could be computed by the sum of the weights of the sub-polygons from the lowest level (level 3) until the level "*N-2*". For example the triangulated polygon in figure 4.14.e has a sub-polygon with weight $w1$ which is computed on the level "*N-3*" and weights $w9$, $w10$ which are computed on the level "*N-4*".

- The last level "*N*" is for finding the first triangle on the initial polygon. Different possibilities are shown by figures 4.14.b - 4.14.d. The configuration with the minimal weight is chosen from all these possibilities. The configuration in figure 4.14.b has a weight of sum of $w0$, $w1$ and $w2$. The $w0$ is computed from the level "*N-3*" whereas the other two weights are computed from the level "*N-4*"'. The weight of the configuration in figure 4.14.d is computed by the sum of the $w6$ which is computed on the level "*N-1*" and the $w7$ which is computed from the level 3.

Once we arrive at the top level we could triangulate the polygon from the top to bottom. That means we begin with the initial polygon and we chose the first triangulation chosen with minimal weight at the level "N". Then for each sub-polygon we will continue to choose the triangulation on different level inferior.

The triangulation proposed in [60] chooses the configuration of minimal area, i.e. the manipulated weights are the areas of the triangles. This choice is not really adapted to planar meshes because the planar surfaces will have the same area whatever the triangulation. Mechanical models often consist of many planar surfaces, which is not very usual in animation models for which the minimal area has been considered. Therefore in this thesis for better fitting the FE mesh modification, the criterion of the weight is changed. The weight we use on each triangle will be the aspect ratio which is introduced in section 4.3.1. Then, the triangulation of each level will chose the combination of the sub-polygons with a sum of weight maximal.

Figure 4.15 illustrates an example of triangulation by using two different criteria. Figure 4.15.a shows an initial triangle mesh on which an empty space is available to triangulate. The triangulation using the minimal area [60] is performed on the model and the result is shown on figure 4.15.b. Figure 4.15.c illustrates the triangulation using the maximal aspect ratio criterion. As de-

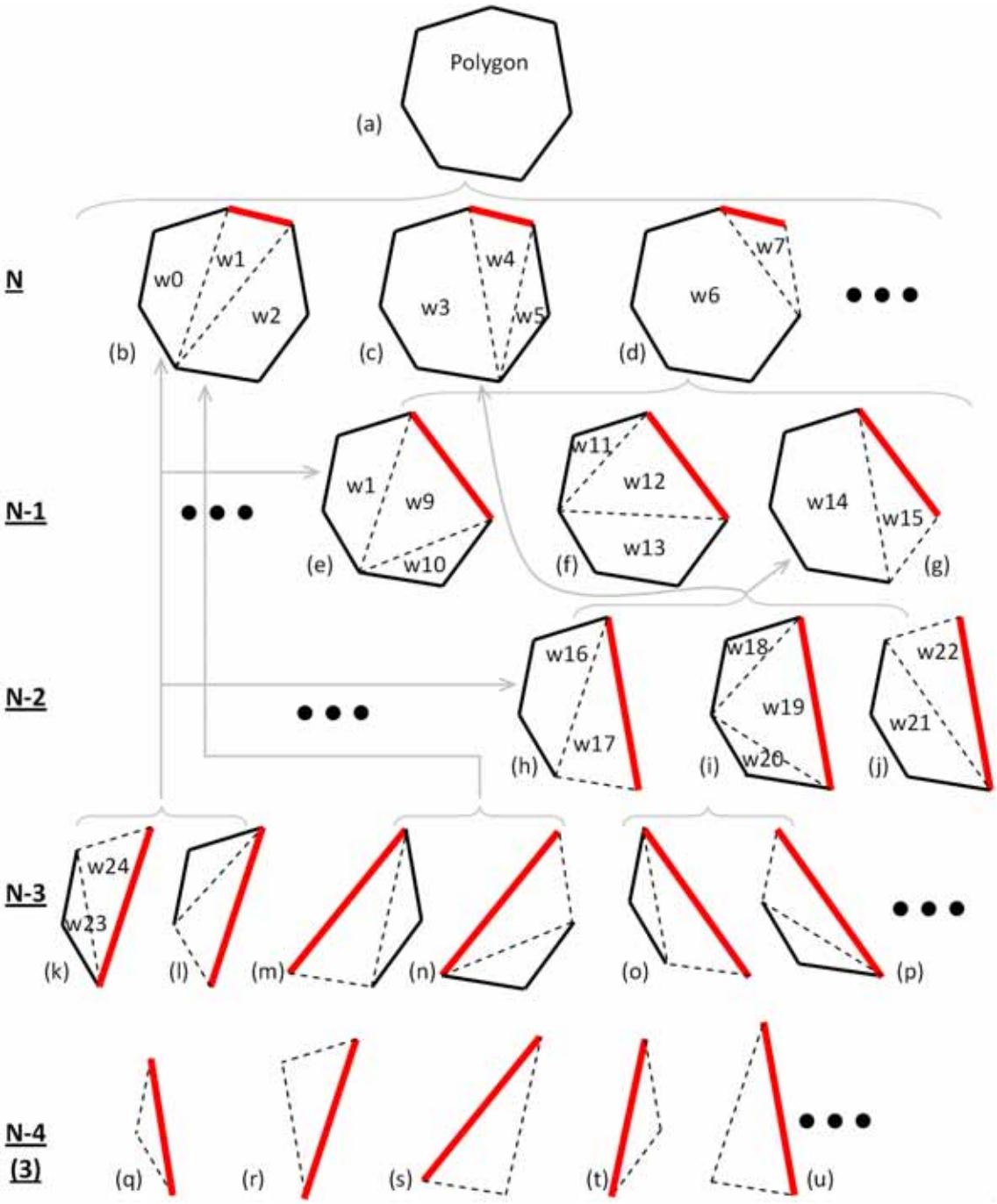


Figure 4.14: Hierarchical triangulation

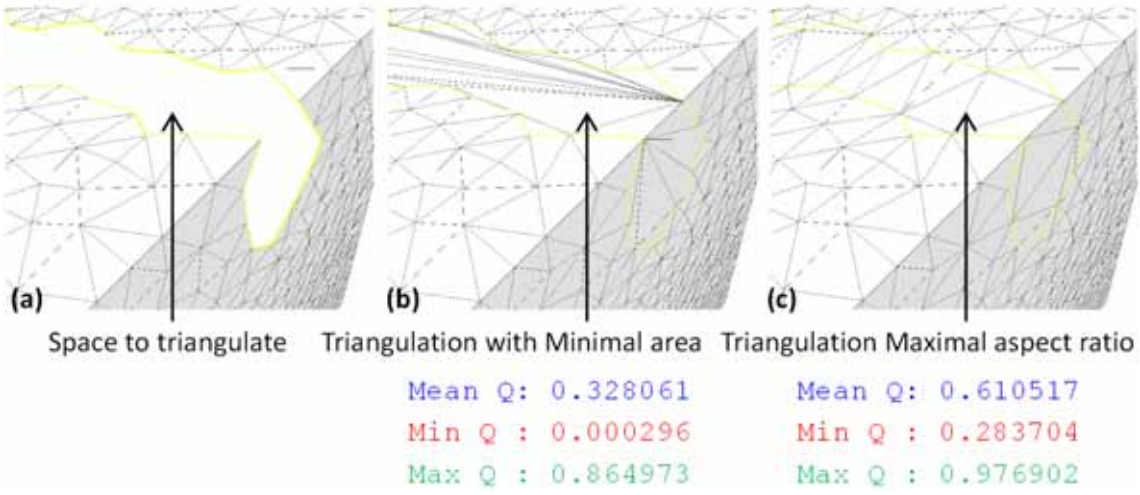


Figure 4.15: (a) Triangle mesh with a hole; (b) triangulation by minimising the area; (c) triangulation by maximising the aspect ratio

picted, when the surface is flat, the triangulation using the minimal area does not give a good result in terms of aspect ratio of the triangles. In figure the mean, minimal and maximal aspect ratio Q is computed on the created patch.

4.4.2 Tetrahedralisation in closed triangle surface

The tetrahedralisation by using a closed triangle surface consists in creating tetrahedra to fill in a space enclosed by a set of triangles. The state-of-the-art of these techniques is presented in chapter 2 (p.48). For prototyping the proposed CAD-less mesh modelling operator, the adopted tetrahedral meshing module is the open-source software package Tetgen [105]. From implementation point of view, it is very easy to use as an external program called by the CAD-less mesh modelling operator. The Tetgen program is used in a constrained-Delaunay mode so that it adds points to the volume interior but it does not add points to the bounding surface. The Delaunay tetrahedralisation (DT) uses the constrained DT method proposed in [106]. The Tetgen software includes a Delaunay refinement approach [103] for ensuring mesh quality, as measured by edge-radius ratios.

An example of tetrahedral mesh generated by Tetgen is shown in figure 4.16.

The original surface mesh has a mean quality (i.e. an aspect ratio Q) of 0.76 and the tetrahedra generated inside have a mean quality of 0.65.

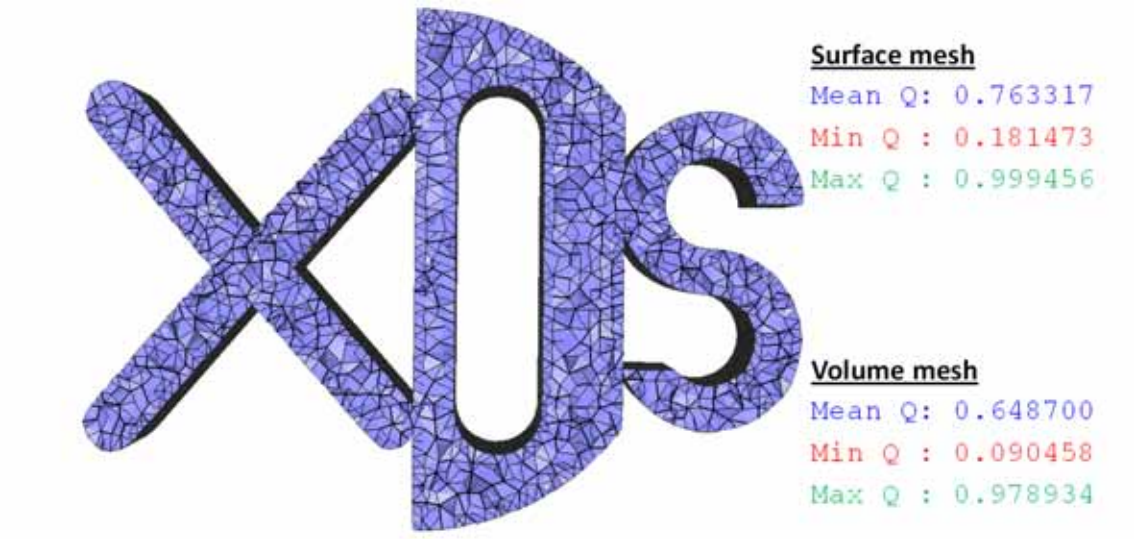


Figure 4.16: Example of tetrahedral mesh generated from a surface mesh by Tetgen [105]

4.4.3 Duplication

In 2D triangle mesh, the so-called "duplication" consists in duplicating a set of edges for dissociating two sub-meshes of the surface. In 3D tetrahedral mesh, the duplication consists in duplicating a set of triangles for dissociating two sub-meshes of the volume. When edges of a 2D mesh, and triangles of a 3D mesh, are duplicated the associated mesh elements of lower dimension might be also duplicated.

When performing mesh entities duplicating several rules have to be taken into account:

- **Rule 1**: The domain of the space occupied by the mesh should not be changed. Therefore, only mesh elements with dimension lower than the space dimension could be duplicated. More precisely, in a 3D volume mesh the 3D mesh elements (ex. tetrahedra) should not be duplicated and in a

2D surface mesh the 2D mesh elements (ex. triangles) should not be duplicated. The conformity of the mesh should be insured during the duplication process. The conformity is not respected when non-manifold elements are generated and the following rules are proposed.

- **Rule 2**: The duplication should start from the higher dimension elements (parent) and the associated lower dimension elements (child) might be duplicated to insure the conformity. Any child elements should be not duplicated without any parent elements duplication.

- **Rule 3**: The duplication of mesh entities will change their position proprieties (interior, boundary and standalone). Parent element duplication will not always lead to child duplication. But a child that situated on the model boundary parent before its duplication should be duplicated for avoiding non-manifold elements.

- **Rule 4**: Always for the mesh conformity aspect, the mesh elements should not be duplicated to produce more instances than the associated parents.

- **Rule 5**: Always for the mesh conformity aspect, when one child is duplicated in to more instances, each parent who is not duplicated should be associated with some of child instances.

Figure 4.17 shows some examples of surface mesh duplication with positive and negative points. The initial mesh is shown in figure 4.17.a, boundary and interior nodes/edges are distinguishable from colors and shape. Nodes n_1 and n_4 are on the boundary whereas nodes n_2 and n_3 are in the interior. Edges represented by continuous segments are boundary edges whereas the dashed segments represent the interior edges. The upper duplication examples (fig.4.17.b) are not allowed whereas the lower ones (fig.4.17.c and fig.4.17.d) are correct. Note that all duplicated mesh elements illustrated in figure are not moved or deformed, the visual shifting between duplication instances and the visual deformation are performed for better seeing several elements that are overlapping in the reality.

The rule 1 is respected since none of triangles is duplicated. The other rules concerning the mesh conformity are more or less broken. A detailed analysis of

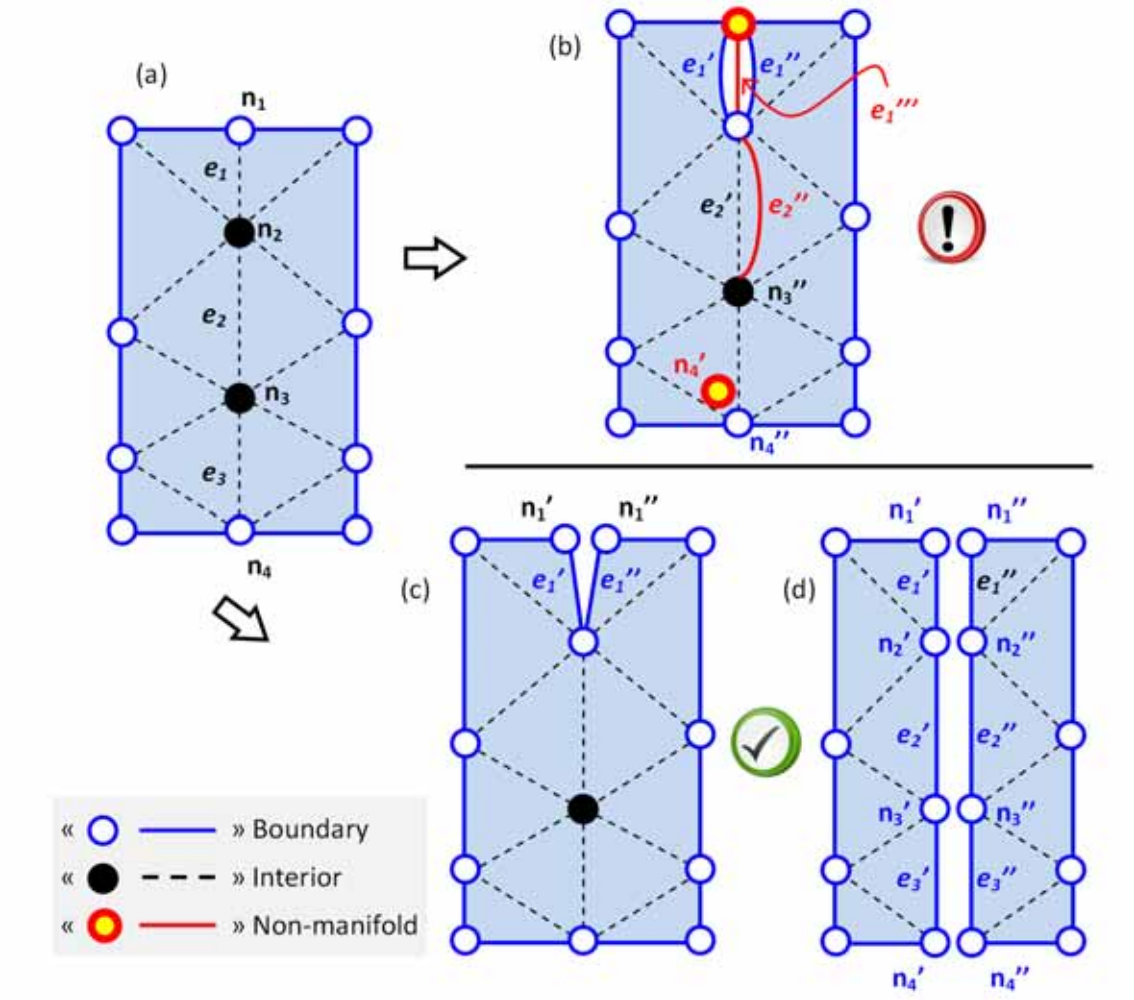


Figure 4.17: Duplications of 2D mesh elements

this aspect is in the following. In the example of figure 4.17.b, the non-manifold elements are identified and are not favorable for the mesh conformity:

- The node n4 is duplicated into two nodes n'_4 and n''_4 without any duplication of parent edges. The node n'_4 becomes a standalone node which is non-manifold. This case breaks the rule 2. Solution: This node should not be duplicated.

- The edge e2 is duplicated into two edges e'_2 and e''_2, but the edge e''_2 does not associate with any parent triangles that was associated with the initial edge

e_2. This case breaks the rule 4. Solution: this edge should not be duplicated as shown in figure 4.17.c or each duplicated edge should associate to a parent triangle as shown in figure 4.17.d.

- The edge e_1 is duplicated into three edges e_1', e_1'' and e_1''', but there were only two parent triangles associated with e_2, therefore this case breaks the rule 4. The solution is to limit the duplication instances into two.

- If the edge e_1 is duplicated into two edges e_1' and e_1'' as described in the last point, the child nodes will change its position propriety: the node n_2 becomes a boundary node and the node n_1 becomes a non-manifold node. The last breaks the rule 3. The solution is to duplicate the node n_2 that was already on the model boundary as shown in figure 4.17.c. The node n_2 becomes boundary type and does not need to be duplicated. But if later on another parent edge of n_2 is duplicated, this node should also be duplicated. It is the case shown in figure 4.17.d where the original edge e_2 is duplicated into two edges e_2' and e_2''.

In FEA, the dissociation between sub-meshes changes definitively the simulation result. It can be used to insert crack or to model a contact between two parts (see section 6.3 on page 202)

4.5 Mesh deformation

The mesh deformation process consists in repositioning the nodes in a mesh for achieving several objectives:

- Relaxing mesh parts for maximising the aspect ratio on the mesh entities;
- Preserve the blending around a junction between mesh parts to be merged;
- Obtaining a certain shape behaviour on a part of mesh.

The relaxation allows repositioning the nodes so that the mesh entities are as much equilateral as possible. Figures 4.18.a and 4.18.b show an example [76] for merging two intersecting triangular meshes “m1” and “m2”, the blue nodes on

the lower mesh "m2" are created by the re-meshing process. The blue nodes in figure [88].a are relaxed by deformation and the final positions shown in figure 4.18.b. Figure 4.18.c illustrates an example of hole filling in a triangular mesh [89]. The patch is created and joined with the mesh while satisfying tangency and curvature blending conditions using mesh deformation (fig.4.18.d). The use of the deformation for reaching the third objective is exemplified in figures 4.18.e and 4.18.f. A part of the boundary of the mesh having a zigzag shape (fig.4.18.e) is transformed into a circle shape (fig.4.18.f). The red nodes are repositioned onto the circle; the green and blue nodes are repositioned on the model so that the mesh quality is maximised [73]. To achieve these three objectives, we use a simple mechanical model named the Force Density Method (**FDM**) [88] that has been modified to fit the requirement of our CAD-less mesh modelling operator.

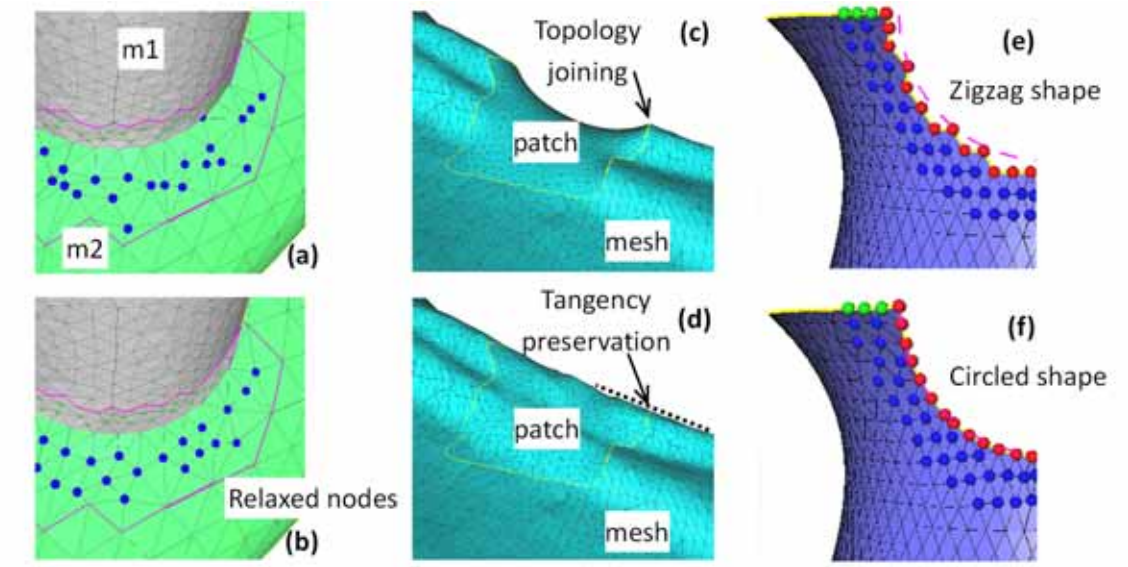


Figure 4.18: Different utilities of mesh deformation (a, b) mesh relaxation; (c, d) curvature preservation on patches junction; (e, f) circle shape rendering from zigzag shape

The adopted deformation technique [88] is presented in details in the following sub-sections. First, a mechanical model coupled with the mesh model (section 4.5.1) as well as its formalisation (section 4.5.2) are introduced. Then, the way the shape can be constrained (section 4.5.4), the way the optimisation problem is set up and solved (section 4.5.3) as well as the different minimisations (section

4.5.5) are discussed.

For prototyping the CAD-less mesh modelling operator, not only the deformation tool is plugged in but also some new functionalities are designed and implemented into the deformation tool during this thesis. The main contribution to the deformation tool itself in the framework of the thesis concerns new deformation constraints introduced in section 4.5.6.

4.5.1 The adopted mechanical model

Basically, the shape of a mesh model is defined by the relative positioning of all the vertices. For being able to change the shape of a mesh, it is necessary to interact simultaneously on the position of all nodes. If the nodes are repositioned sequentially one after another, the post-positioned nodes do not affect the position of fore-positioned nodes. Otherwise a loop for repeating the sequential positioning should be performed, which may need more time. In addition, it is very difficult to understand how a node can react to a repositioning of another node faraway in terms of connectivity. Therefore the repositioning of nodes should be considered more globally and interactively. The point is how the position of a node influences on the other nodes in terms of quantity and orientation.

The proposition is to act on the positions of two nodes that are connected by an edge. For being able to define the quantity of the relative position influence between two nodes connected by an edge, the length of the edge is taken into account. For covering all these needs the authors of [88] propose to couple a simple mechanical model, a network of bars, onto the mesh model. The vertices and edges of the meshes match respectively the nodes and the bars of a network (fig.4.19). Each bar i associating to a node n can be seen as a spring with a null initial length and a stiffness qi (more precisely a force density). To preserve the static equilibrium state of the structure, external forces fn have to be applied to the nodes n. If these external forces were not applied to the network, all the nodes would be gathered together at a single point. The linear relationships between the external forces and the position of the nodes enable intuitive shape modifications through the manipulation of a restricted set of external forces. Therefore the repositioning of a node could influence directly the other nodes sharing the same

edges due to the changes of the spring forces. For keeping the equilibrium the directly influenced nodes will react to adapt the situation and change its position, which will influence the other nodes reachable by edges. The global reaction of repositioning from one node to another node is finally handled through the use of the static equilibrium equations of all the nodes.

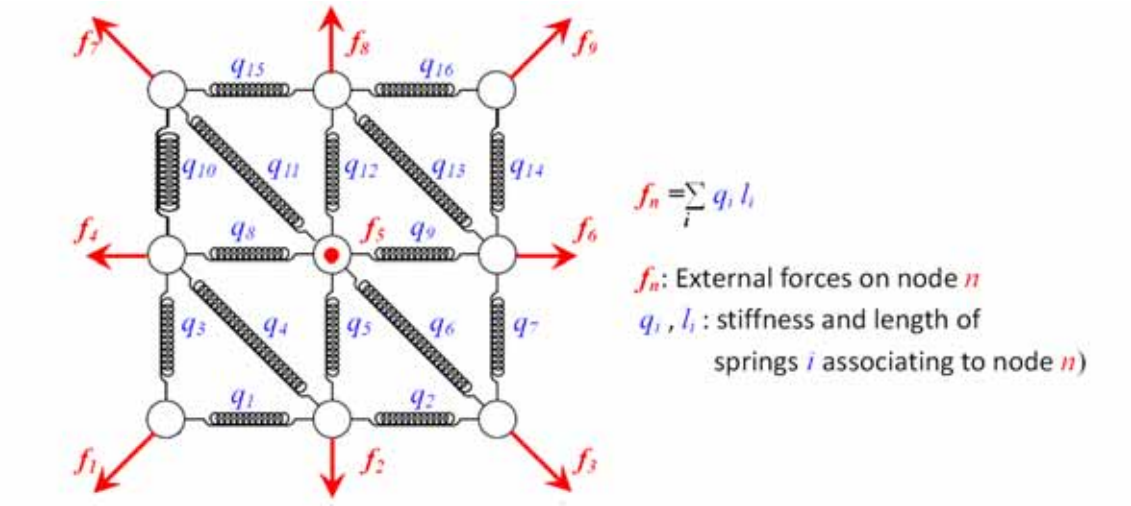


Figure 4.19: Network of springs associated to geometric model

The difference between the manipulation of pure geometry and geometry with FDM is clearly shown by the simple example in figure 4.20. Starting from a mesh model (fig.4.20.a) the external-ring nodes are repositioned far away from the barycentre (fig.4.20.b). If the FDM method is associated with geometry model, the initial equilibrium state of the model should be maintained by applying external forces on the external ring nodes (fig.4.20.c). The external nodes are repositioned far away from the barycentre, if the internal ring nodes are blocked to move there should be certain external forces applied on these blocked nodes (fig.4.20.d). If these internal ring nodes are free to move and if no external forces applied on them, these nodes should move to adapt a new equilibrium state (fig.4.20.e). At the same time the external forces applied on the external ring nodes reduce also.

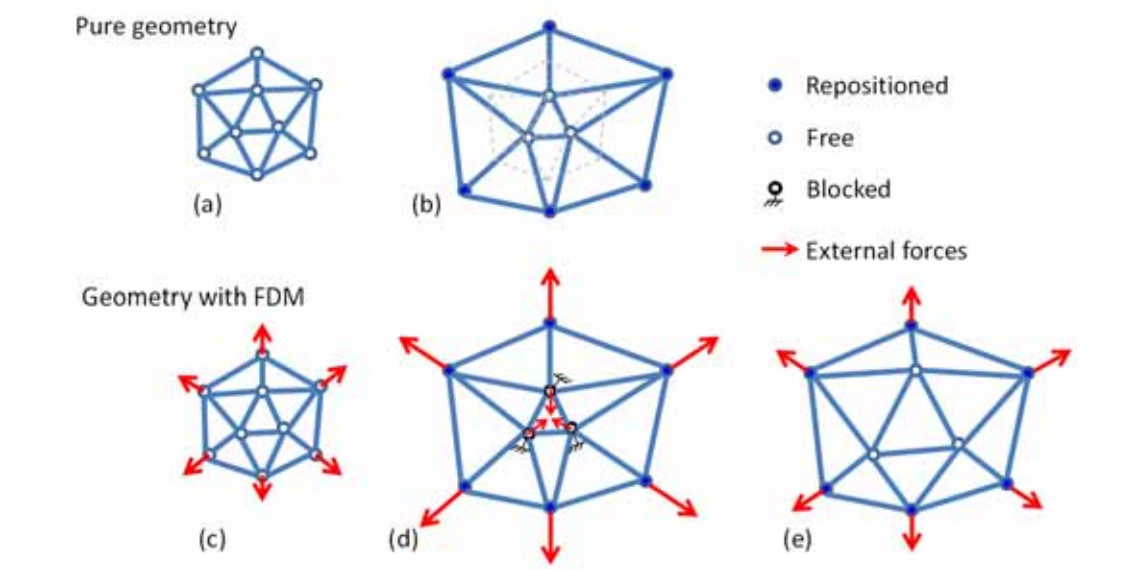


Figure 4.20: Repositioning of nodes on a mesh (a, b) and on a mesh with FDM (force density method)

4.5.2 Force Density Method formalisation

A bar network coupled with a mesh model has its nodes equivalent to the vertices of the mesh in terms of position and numbering. The bars are also synchronised with the edges of the mesh in terms of connectivity with the vertices, length and numbering. Let N_b be the number of bars and let N_n be the number of nodes for a bar network.

For expressing the connectivity of a bar network a single branch-node matrix C of size $(N_b \times N_n)$ is created as following:

$$C_{i,j} = \begin{cases} 1 & j^{th}\ node\ is\ the\ 1^{st}\ extremity\ of\ i^{th}\ bar \\ -1 & j^{th}\ node\ is\ the\ 2^{nd}\ extremity\ of\ i^{th}\ bar \\ 0 & otherwise \end{cases} \tag{4.39}$$

A force density matrix Q of size $(N_v \times N_n)$ is also defined by the following formula:

$$Q_{i,j} = q_j \cdot \delta \ \ (where\ \delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & j \neq j \end{cases}) \tag{4.40}$$

where $\delta_{i,j}$ is the Kroenecker's symbol and $q_j = f_j/\ell_j$ is the force density into the j^{th} bar of length ℓ_j

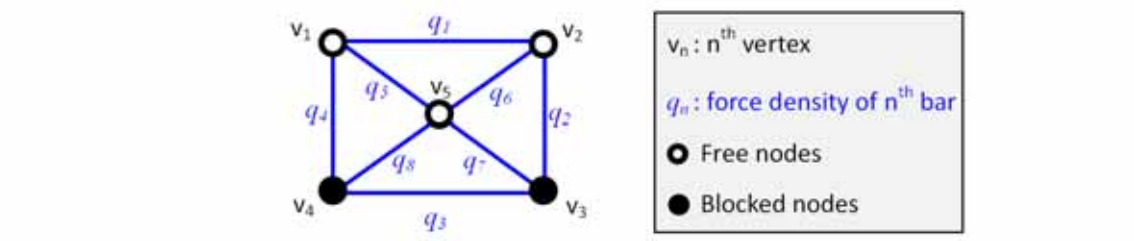


Figure 4.21: Example of bar network

The bar network shown in figure 4.21 is created from a mesh and its vertices are numbered as well as the force densities in the bars. There are 2 blocked nodes and 3 free nodes. The connectivity matrix C and the force density matrix Q for this bar network (fig.4.21) is:

$$C_{i,j} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} Q_{i,j} = \begin{bmatrix} q_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & q_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_8 \end{bmatrix} \tag{4.41}$$

Let N_{fn} and N_{bn} be separately the number of free nodes and blocked nodes. Therefore the connectivity matrix C can be decomposed into two matrices C_{fn} $(N_b \times N_{fn})$ and C_{bn} $(N_b N_{bn})$ representing the connections related respectively to the N_{fn} free nodes and the N_{bn} blocked nodes. Therefore the connectivity matrix

for the bar network shown in figure 4.21 could be written in the following way:

$$
\begin{array}{l}
\textit{node no.} \quad 3^\circ\ 4^\circ \quad 1^\circ\ 2^\circ\ 5^\circ \\
\qquad\qquad\ \downarrow\ \downarrow \quad\ \downarrow\ \downarrow\ \downarrow \\
C = \left[\begin{array}{cc|ccc}
0 & 0 & 1 & -1 & 0 \\
-1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & -1 \\
0 & 0 & -1 & 0 & 1 \\
0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 1 & -1 \\
1 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & -1
\end{array}\right] = [C_{bn} \,|\, C_{fn}] \\
\qquad\qquad\ \underbrace{\qquad}_{blocked} \quad \underbrace{\qquad\qquad}_{free}
\end{array}
\tag{4.42}
$$

Given X, Y and Z, the three vectors containing the components of the 3D coordinates of the N_v nodes of the bar network coupled to the mesh nodes, the F_x , F_y and F_z components of the external forces applied to these nodes can be obtained by using the following $(3 \times N_n)$ equations expressing the bar network static equilibrium:

$$
\begin{aligned}
F_x &= ({}^tC \cdot Q \cdot C) \cdot X \\
F_y &= ({}^tC \cdot Q \cdot C) \cdot Y \\
F_z &= ({}^tC \cdot Q \cdot C) \cdot Z
\end{aligned}
\tag{4.43}
$$

For expressing the force density between any pair of two nodes whatever the connectivity in between, a single matrix D is created for merging the force density matrix Q and the network connectivity matrix C. The D matrices are obtained through the decompositions of the ${}^tC \cdot Q \cdot C$ matrix. The matrix D is not directly used but decomposed into sub-matrices with respect to the free and blocked nodes $(D_f, D_b, D_{bf}, {}^t D_{bf})$:

$$
\begin{array}{c}
\left[\begin{array}{ccc} {}^tC & Q & C \end{array}\right] = \left[\begin{array}{cc} {}^tC_{fn} & {}^tC_{bn} \end{array}\right] Q^t \left[\begin{array}{c} {}^tC_{fn} \\ {}^tC_{bn} \end{array}\right] = \\
\left[\begin{array}{ccc|ccc}
{}^tC_{fn} & Q & C_{fn} & {}^tC_{fn} & Q & C_{bn} \\
{}^tC_{bn} & Q & C_{fn} & {}^tC_{bn} & Q & C_{bn}
\end{array}\right] = \left[\begin{array}{cc} D_f & D_{bf} \\ {}^tD_{bf} & D_b \end{array}\right]
\end{array}
\tag{4.44}
$$

Therefore a distinction between free (X_{fn}) and blocked (X_{bn}) nodes can be performed in the equations 4.43. It gives rise to the two following sets of equations. One for the external forces applied to free nodes:

$$\begin{aligned} F_{fn_x} &= D_f \cdot X_{fn} + D_{bf} \cdot X_{bn} \\ F_{fn_y} &= D_f \cdot Y_{fn} + D_{bf} \cdot Y_{bn} \\ F_{fn_z} &= D_f \cdot Z_{fn} + D_{bf} \cdot Z_{bn} \end{aligned} \tag{4.45}$$

And another for the external forces applied at blocked nodes:

$$\begin{aligned} F_{bn_x} &= {}^tD_{bf} \cdot X_{fn} + D_b \cdot X_{bn} \\ F_{bn_y} &= {}^tD_{bf} \cdot Y_{fn} + D_b \cdot Y_{bn} \\ F_{bn_z} &= {}^tD_{bf} \cdot Z_{fn} + D_b \cdot Z_{bn} \end{aligned} \tag{4.46}$$

Conversely, being given a set of external forces applied to the nodes of the bar network, the positions of the free nodes can be found from the equation 4.45 and are given by:

$$\begin{aligned} X_{fn_x} &= (D_f)^{-1} \cdot (F_{fn_x} - D_{bf} \cdot X_{bn}) \\ X_{fn_y} &= (D_f)^{-1} \cdot (F_{fn_y} - D_{bf} \cdot Y_{bn}) \\ X_{fn_z} &= (D_f)^{-1} \cdot (F_{fn_z} - D_{bf} \cdot Z_{bn}) \end{aligned} \tag{4.47}$$

These last equations show how it is possible to manipulate indirectly the node positions, and consequently the inner vertices, through the manipulation of external forces (see [88], for the treatment of configurations where the D_f matrix is singular). By giving all the positions of the nodes, the external forces applied on the free nodes as well as blocked nodes can be computed using equations 4.45 and 4.46. By giving the different external forces on the nodes, the positions of the free nodes can be determined using equation 4.47. Therefore, the unknowns of the deformation process could either be the positions or the external forces themselves.

4.5.3 Optimisation problem formulation

To achieve mesh deformation, an optimisation problem is solved. It is defined by a set of linear and non-linear equality constraints and an objective function ϕ to

be minimised (eq.4.48). The unknowns are either the positions of the mesh nodes or the external forces applied to these nodes. They are gathered together in the unknown vector $U(u_1\, u_2\, \cdots, u_{Nu})$ of size $N_u = 3 \times N_n$, that could be either the nodes positions vector $X(x_1, x_2, \cdots, x_{Nn}, y_1, y_2, \cdots, y_{Nn}, z_1, z_2, \cdots, z_{Nn})$ of size $3\times N_n$ or the external forces $F(f_1^x, f_2^x, \cdots, f_{Nn}^x, f_1^y, f_2^y, \cdots, f_{Nn}^y, f_1^z, f_2^z, \cdots, f_{Nn}^z)$ of size $3 \times N_n$. The constraint vector G contains different constraints $(g_1, g_2 \cdots g_{Ng})$ of size N_g that will be detailed in the next subsection. The different objective function to be minimised will be detailed in the subsection 4.5.5. Therefore the optimisation could be summarised as follows:

$$\begin{cases} G(U) = 0 \\ \min \phi(U) \end{cases} \tag{4.48}$$

Here, the objective functions to be minimised are always quadratic, e.g. the length of the bars, the the norm of the external forces and so on. Any linear combination of these quadratic functions of the unknown vector U can be written into the following expression:

$$\phi(U) = \frac{1}{2} U^T C U - D^T U + E \tag{4.49}$$

where the C and D matrices are the quadratic and linear coefficients and E is the constant matrix. The minimum of this quadratic function ϕ is given by solving the following equations system:

$$CU = D \tag{4.50}$$

The constrained optimisation problems are solved by using the Lagrangian multipliers to integrate the constraints into the objective function to minimise [88]. The computation of the gradient of the constraints vector may be difficult depending on whether the constraints are linear or not. If some of the equations of the constraints vector are non-linear, the resolution process becomes iterative. When the constraints are non-linear, they are linearised at the first order.

Figure 4.22 shows an example of displacement of nodes on a mesh. Figures 4.22.c and 4.22.d shows the nodes that have to be moved and the destinations

of those nodes. The triangles associating with these nodes to move are colored into green. If only the pure geometry is manipulated, the nodes are repositioned simply (fig.4.22.a and fig.4.22.b). Whereas, when the mesh model is coupled with the FDM method (fig.4.22.e and fig.4.22.f), the nodes to move are also moved to the destinations but the other nodes are also moved to maintain the static equilibrium state.

The different objective functions to minimise and the various constraints are detailed in the next sub-sections.

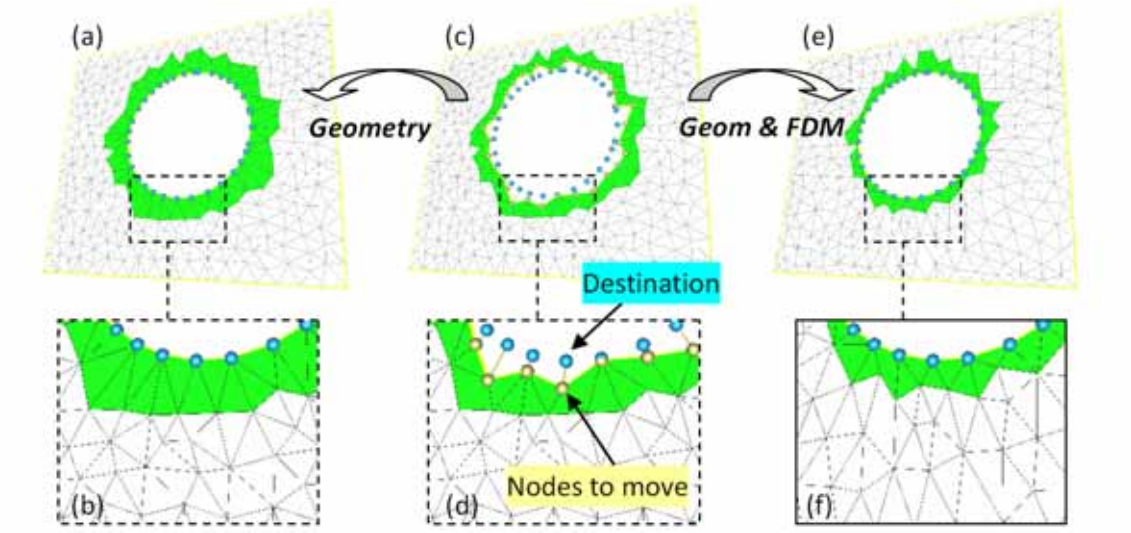


Figure 4.22: Nodal displacement definitions (c, d), displacement of pure mesh (a, b) and displacement of mesh coupled with a FDM mechanical model (e, f)

4.5.4 Shape constraint formalisation

To be able to define constraints over a triangle mesh, let us first introduce the way a so-called parametric point can be located on a mesh:

$$\begin{aligned} P_p^x &= \omega_a \cdot X_a + \omega_b \cdot X_b + \omega_c \cdot X_c \\ P_p^y &= \omega_a \cdot X_{N_n+a} + \omega_b \cdot X_{N_n+b} + \omega_c \cdot X_{N_n+c} \\ P_p^z &= \omega_a \cdot X_{2N_n+a} + \omega_b \cdot X_{2N_n+b} + \omega_c \cdot X_{2N_n+c} \end{aligned} \tag{4.51}$$

where X is the vector containing successively the x, y and z coordinates of the Nn free nodes (see previous section). The numbers a, b and c correspond to the id of the three nodes defining the triangle on which the parametric point lie. The parameters ω_a, ω_b and ω_c are the barycentric weights to be able to retrieve the 3D coordinates (P_p^x, P_p^y, P_p^z) of the parametric point from the coordinates of the three vertices of the triangle on which it lies.

Two types of constraints can be defined: geometric and parametric constraints. The geometric constraints are defined between a parametric point P_p and a geometric point P_g, whereas the parametric constraints are defined between two different parametric points P_p^1 and P_p^2.

From a practical point of view, when a node of a mesh should move to a new position, in this case a geometric point P_g is created for taking the new position and a parametric point P_p is created for the mesh node. Then, a geometric position constraint is defined between the geometric and parametric points such that:

$$g_{position}(U) = \|P_p - P_g\| = 0 \tag{4.52}$$

where the P_p is computed from the position vector with equation 4.51 and U the unknown vector is the positions vector X or the forces vector F which could be computed from the positions vector X as described previously.

Similarly, when a mesh node should coincide with another mesh node, two parametric points P_p^1 and P_p^2 are created for the two concerned mesh nodes. A parametric coincidence constraint is defined with the two parametric points such that:

$$g_{coincidence}(U) = \left\|P_p^1 - P_p^2\right\| = 0 \tag{4.53}$$

where the P_p^1 and P_p^2 are computed from the unknown vector U by using the equation 4.51. More constraints are developed in this PhD thesis and presented in the subsection 4.5.6.

4.5.5 Different minimisations

For deforming the geometric model coupled with a bar network, the objective function ϕ to minimise (eq.4.48) should be detailed. Among the various combi-

nations that can be used, the simplest one is the minimisation of the **sum of the external forces**:

$$\phi(U) = \sum_i F_i^2 = \sum_i \left((f_i^x)^2 + (f_i^y)^2 + (f_i^z)^2\right) \tag{4.54}$$

Depending on whether the unknowns are the positions X or the forces F, the equations 4.45 have to be used. The sum of the external forces can be done over the free nodes or over both the free and blocked nodes. The minimisation of the sum of the external forces applied on the free nodes can be written into:

$$\phi(X) = \sum_{free} \|F_{fn}\|^2 \tag{4.55}$$

Since the bars connected with blocked nodes that are surrounded by blocked nodes will not change the length therefore the external forces applied on these blocked nodes will not change. As consequence the minimisation of external forces on the blocked nodes concerns only the ones applied on the 1^{st} blocked nodes that could reach at least one free node through a bar. The minimisation of the sum of the external forces applied on free and first-line blocked nodes could be written as:

$$\phi(X) = \sum_{free} \|F_{fn}\|^2 + \sum_{1^{st}blocked} \|F_{bn}\|^2 \tag{4.56}$$

As it will be presented later, the choice of the minimisation may strongly affect the result of the deformation. The way the blocked nodes are taken into account also affects the results.

Beside the sum of external forces, the **sum of the variations of the external forces** from one node to another is also useful. In this case the objective function will be the sum of the difference between the forces applied on a node i and on its neighbours j. Therefore the objective function relative to the unknown nodes i and their neighbours j can be written into:

$$\phi(U) = \sum_i \sum_j \|F_i - F_j\|^2 = \sum_i \sum_j \left((f_i^x - f_i^x)^2 + (f_i^y - f_i^y)^2 + (f_i^z - f_i^z)^2\right) \tag{4.57}$$

Similarly this variation of external forces could be also applied only on the free nodes or on both free and blocked nodes. The objective function for expressing the variation of external forces applied only on free nodes is:

$$\phi(X) = \sum_{i=free} \sum_{j} \|F_i - F_j\|^2 \tag{4.58}$$

and the one expressing the variation of external forces applied both on free and blocked nodes is:

$$\phi(X) = \sum_{i=free} \sum_{j} \|F_i - F_j\|^2 + \sum_{i=1^{st}blocked} \sum_{j} \|F_i - F_j\|^2 \tag{4.59}$$

Figure 4.23 shows an example with four different minimisations. The free nodes are the ones enclosed within the last one free nodes series. The blocked nodes are therefore the rest nodes among which the 1^{st} line blocked nodes are in connection directly with the last line free nodes. The lines not on the mesh model shows the external forces (orientation, length) applied on each node.

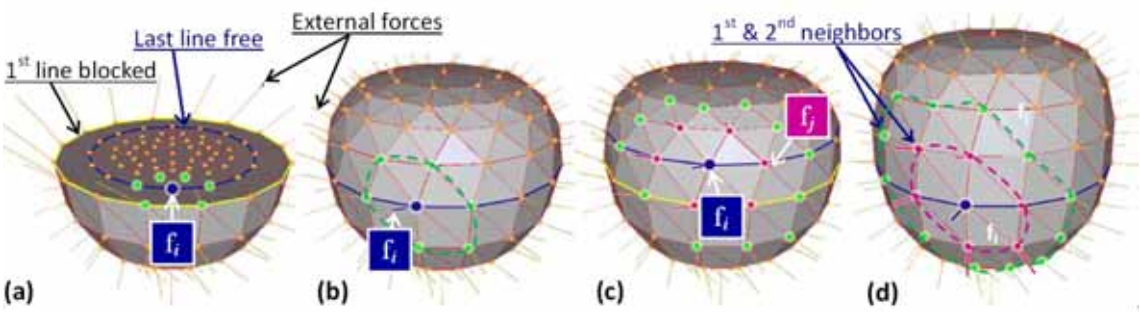


Figure 4.23: minimisation of external forces only applied on the free nodes (a), minimisation of external forces applied on the free and blocked nodes (b), minimisation of external force variation on free nodes (c) and minimisation of external forces variation on free and blocked nodes (d) [89].

Figure 4.23.a shows the minimisation of the sum of the external forces applied on the free nodes, which corresponds to the equation 4.55. Since the external forces applied to the free nodes are independent from each other, the minimisation of this sum on all the free nodes always produces a null unknown vector (no crossed terms in the quadratic function). This tends to minimise the curvature

of the underlying geometry while producing areas as planar as possible.

The minimisation of the sum of the external forces applied both on the free nodes and blocked nodes, which corresponds to the equation 4.56, is shown in figure 4.23.b. In this example, the external forces applied on the blocked nodes are also minimised which enables a tangency blending between free nodes and blocked nodes. Both examples (fig.4.23.a and fig.4.23.b) using the quantity of external forces applied on each node, consist in taking into account positions only from nodes of the first neighbourhood.

An example of minimisation of external force variation only applied on the free nodes, which corresponds to the equation 4.58 is shown in figure 4.23.c. The sum of difference between the force f_i applied on a node i and the forces f_j applied on its 1^{st} neighbours j is minimised. In figure 4.23.d the force variations computed both on the free and block nodes are minimised, which corresponds to the equation 4.59.

One can notice that the force variation minimisation tends to get position information from 1^{st} and 2^{nd} neighbours of each concerned node which tends to minimise the curvature variation across the boundary between the blocked and free nodes.

4.5.6 New shape constraints for CAD-less mesh modelling operator

As shown in the equation 4.48, the deformation engine is based on the resolution of an optimisation problem. The constraint vector G gathers together the equations constraining the deformation. Any equations describing a relationship, linear or non-linear, between the positions of the vertices in the deformed area are possible. Thus, in this work, we extend the capabilities of the method to cover most of the needs in mechanical engineering. Notably, the basic shape constraints, such as planar, spherical and cylindrical constraints have been considered and are hereafter detailed.

As presented previously the constraints can be of two types: geometric and parametric. The newly created constraints all belong to the category of geometric constraints. Therefore, a geometric point P_g is created to represent a 3D point of

coordinates (x_g, y_g, z_g). A parametric point P_p of coordinates (x_p, y_p, z_p) is also created and located on the mesh while using the equation 4.51. In addition, n is a unit normal vector of components (n_x, n_y, n_z).

As introduced in the previous section, when the constraints are non-linear, a linearisation at the first order is performed for the various constraints which are hereafter detailed:

- **planar constraint**: The parametric point P_p has to stay on a plane defined by the point P_g and the normal n. The parametric constraint equality of type "planar constraint" g_{plane} is expressed as:

$$g_{plane}(U) = (P_p - P_g) \cdot n = 0 \tag{4.60}$$

  Here, there is just one scalar equation that depends linearly of the position of P_p.

- **spherical constraint**: The parametric point P_p has to stay on a sphere centered in P_g and with a radius R. The parametric constraint equality of type "sphere constraint" g_{sphere} is expressed as:

$$g_{sphere}(U) = \|P_p - P_g\|^2 - R^2 = 0 \tag{4.61}$$

  This non-linear scalar equation can be linearised according to the components of the unknown vector U.

- **cylindrical constraint**: The parametric point P_p has to stay on a cylinder that is defined by a geometric point P_g with a radius R and an unit vector n characterising its axis:

$$g_{cylinder}(U) = \|(P_p - P_g) \wedge n\|^2 - R^2 = 0 \tag{4.62}$$

  The cylindrical constraint can be linearised according to the unknown vector U.

- **free-form constraint**: when the shape of the surface where a parametric point P_p should stay on, does not correspond to any of the previously in-

troduced constraints, it is considered as free-form shape. In this case the parametric point P_p has to stay on the tangent plane defined by the position of P_p at the initial iteration and the normal n to the mesh at this point. It becomes the plane constraint presented at beginning:

$$g^{[k]}_{freeform}(U) = ((P_p - P_g) \cdot n^{[k]} = 0 \tag{4.63}$$

An equation similar to 4.60 can easily be obtained.

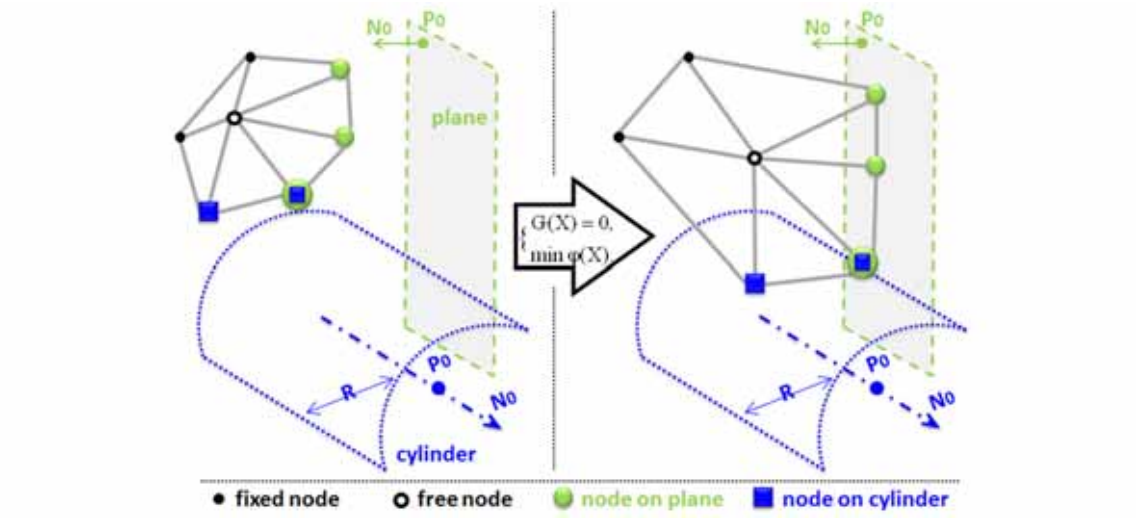


Figure 4.24: Example of deformation using force minimisation under plane and cylinder type constraint

A simple schematic example of mesh adjustment using constrained deformation is shown in figure 4.24. At the left part different boundary conditions are defined on all nodes of the initial mesh. These nodes are labeled in different ways detailed under the pictures. The "fixed nodes" do not move during the deformation. The "free nodes" can move without any constraint in the space. The "nodes on plane" are free to move only on the plane and the "nodes on cylinder" are free to move only on the cylinder. The right picture gives the result of the deformation by applying external forces minimisation under the above shape constraints. The various constraints are satisfied so that some constrained nodes stay either on the plane or on the cylinder or both. The node subject to two

constraints is staying on the intersection zone between the two surfaces. The free node moves to a location where the mesh is relaxed, i.e. the sum of the external forces is minimised. The two fixed nodes remain at their initial positions. The mesh modification shown in figure 4.24.f is using the deformation under several constraints.

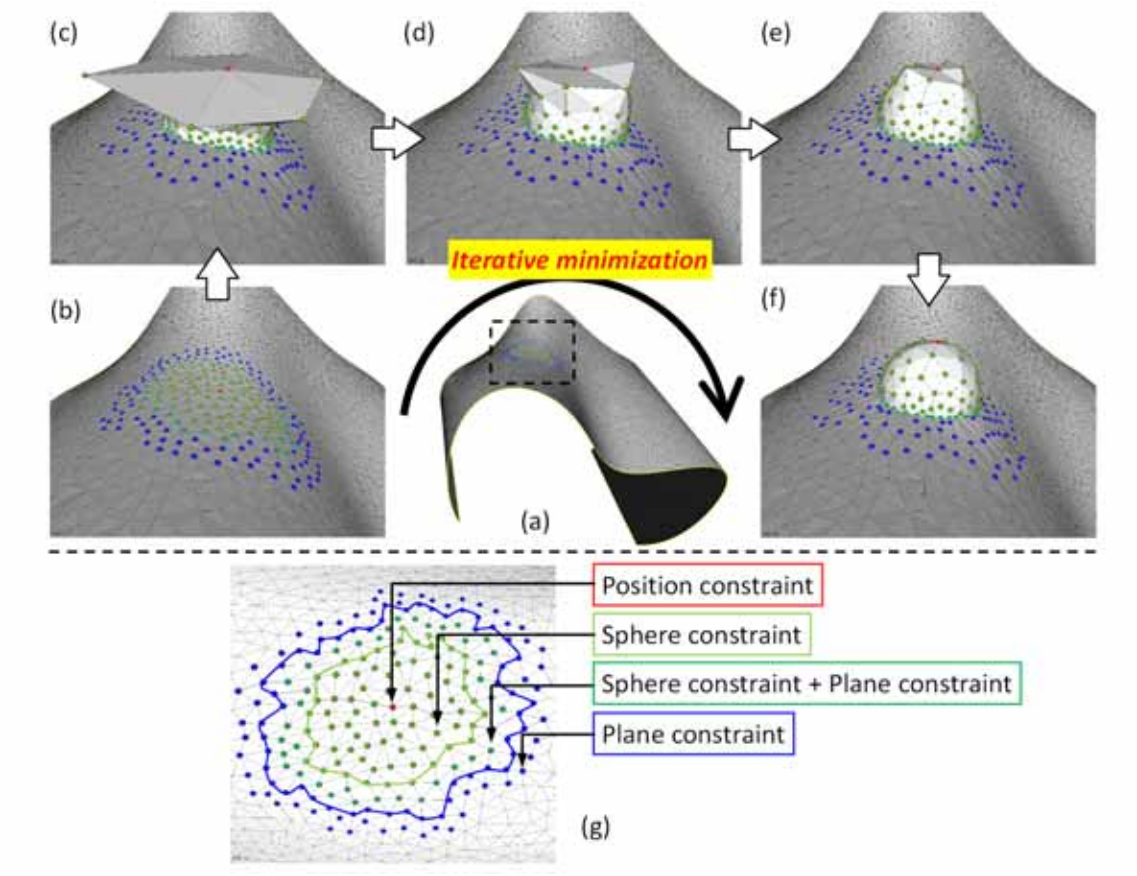


Figure 4.25: Mesh deformation for rendering a sphere shape by iterative minimisation

A real mesh deformation example is presented on figure 4.25. It consists in the deformation of a freeform surface (fig.4.25.a), especially in the framed zone. The objective shape is a sphere that consists in a set of non-linear constraints assigned to several vertices, therefore the minimisation becomes iterative. From figure 4.25.b to figure 4.25.f, different iterations of the resolution are shown, starting from initial state. With 4 iterations, the distances between all nodes and the

sphere are lower than 10^{-3} mm for the mesh part of size $10 \times 10\, mm$. The different constraints assigned on the nodes are shown in figure 4.25.g. The middle red node is constrained to stay on the top position of the sphere whereas the other light and dark green nodes are constrained to stay on the sphere surface. The external ring of the green nodes and the blue nodes are constrained to stay on their tangent plane. All the other nodes without any special color are blocked. The minimisation used is the external forces minimisation applied only on the free nodes, which corresponds to the equation 4.55.

Similar examples could have been presented for the other constraints specifically developed for the CAD-less mesh modelling operator, however, they will be presented in section 6.4 ($p_{.221}$) together with their application on real industrial examples.

4.6 Conclusion

Different basic models, methods and tools have been presented and enable the manipulation of mesh models on geometric level, mainly triangle as well as tetrahedral meshes in the present case. These tools will be used for constructing different FE mesh modification instances of our CAD-less mesh modelling operator and their application on concrete cases will be presented in chapter 6 ($p_{.179}$). Notably, in subsection 4.4.1 ($p_{.121}$), a triangulation technique has been improved to create a surface patch using another criterion for maximising the triangles aspect ratios instead of their areas. It is more suitable for the FE mesh models. Moreover, the used adopted deformation tool that was originally designed to deform surface models (NURBS patch, polyhedral mesh, etc.) has been extended to volume mesh deformation of volume meshes problem. To this aim, new constraints have been proposed and implemented for answering more needs of FE mesh model modifications.

Chapter 5

Basic models, methods and tools to handle semantics

The semantics considered in this thesis involves different kinds of information carried by theFE model. According to the schema of multilayer information (fig.3.1), semantic data are dealt with at the upper layers. At the highest level, semantic data having meaning for a given application are considered, whereas at the middle level it is reflected in collecting mesh entities to allow the association of the application related semantics with the geometry. Since the aim of the operators developed in this thesis work is also to transfer the associated semantics information according to mesh modification, methods for its manipulation have been devised and based on the FE mesh group management. In this chapter, different needs for semantics treatments are discussed. The group manipulation is presented together with the various aspects related to group specification and management: at first, the topology of the group and the relationship between groups then the treatments for preservation of groups are studied. Finally, the semantics analysis and semantics transfer are shown in the end of the chapter.

The semantics transfer during the mesh modification process can involve preservation, transformation and propagation of semantics required for physical FE simulation. The preservation of mesh groups and associated simulation semantics concerns the maintenance of the corresponding description as well as the supporting group definition. While the transformation and propagation of

high-level simulation semantics require also changes of the group on which the semantics is defined by adding other geometry part to define the semantics.

5.1 Basic characteristics of mesh groups

In this section, at first, mesh groups are characterised according to the dimensions of the constituting mesh entities. Then possible topologies of groups are detailed. At the end, the relationships between groups are discussed. For identifying the connectivity of group elements, we consider the connected mesh entities (possibly not belonging to the group) of lower or higher dimension. Then, the spatial relationship between groups is analysed by comparing the "unified dimension" area of the groups.

5.1.1 FE mesh group dimension classification

We have firstly analysed the composition of mesh entities currently used to constitute mesh groups (or low-level semantic data) used to support the FE simulation semantic information (high-level semantic data) in industrial configurations,in particular during the EDF's numerical studies. In all cases, the mesh groups can be distinguished by their constituting mesh elements' dimension as follow:

- groups of nodes (0D):
  - group containing one node,
  - group containing a set of nodes,
- groups of edges (1 dimension):
  - group containing one edge,
  - group containing a set of edges,
- groups of surface elements (2 dimension):
  - group containing one face,
  - group containing a set of faces,

- groups of volume elements (3 dimension):
  - group containing one volume,
  - group containing a set of volumes,
- groups of mixed dimension elements (multiple dimension):
  - group containing a set of 0D - 3D elements.

5.1.2 Topology of mesh groups

The topology of the mesh group is determined by connection relationships of the composing elements and their coverage with respect to the whole mesh. Two mesh elements of the group are **connected** if from one element the other is reachable through the tessellation connectivity of the group elements. In the opposite way two elements of the group are **disconnected**. Thus, the group topology can be classified into 3 categories as following.

Definition 5.1 (**Simply connected** without "hole" inside)**.** *A group is Simply Connected if any two elements of the group are connected as defined above as well as all the elements of the group boundary are connected.*

Figure 5.1.a shows an example of this case: in a mesh model of white 3D cube a "simply connected" group G is defined.

Definition 5.2 (**Connected with "holes"**')**.** *A group is connected with "holes" if all the elements of the group are connected but not all the group boundary elements are connected forming two or more connected subsets of group boundary elements.*

Figure 5.1.b gives an example of 3D group G in which there is a void V. Two connected subsets of the group boundary are pointed. For better understanding, figure 5.1c shows a 2D section view of this model with the group G having a void labeled V. The group G has two subsets of connected group boundary elements.

Definition 5.3 (**Disconnected**)**.** *A group is disconnected if it contains at least two mesh element sets that are not connected. As a consequence the boundary elements are also disconnected.*

An example of this kind of groups is shown in figure 5.1.d. The group G occupies two separate domains that are represented by two blue small cubes. The elements from one group zone can be reached by elements of the other zone only by traversing also elements of the mesh not belonging to the group and the group boundary elements on the left cube are separated from the ones on the right cube.

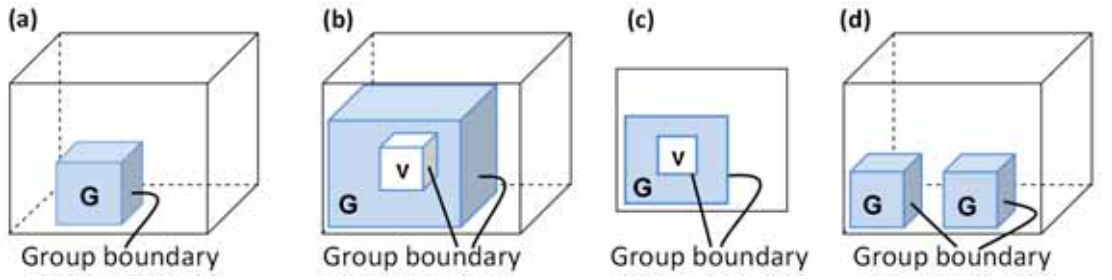


Figure 5.1: Different topologies of group: (a) simply connected (b, c) connected with holes and (d) disconnected

5.1.3 Relationships between mesh groups

In this section, an analysis of possible spatial relationships between two groups is presented; it is based on a spatial intersection between the two groups. The possible relationships identified are:

- **Disconnection**: the intersection between the two groups is null
- **Adjacency**: the intersection between the two groups is only on the boundary
- **Intersection**: the intersection between the two groups is only a part of group for both groups
  - **Interior intersection**: the intersection part does not concern at all the group boundary
  - **Adjacent intersection**: the intersection concerns a part of group boundary

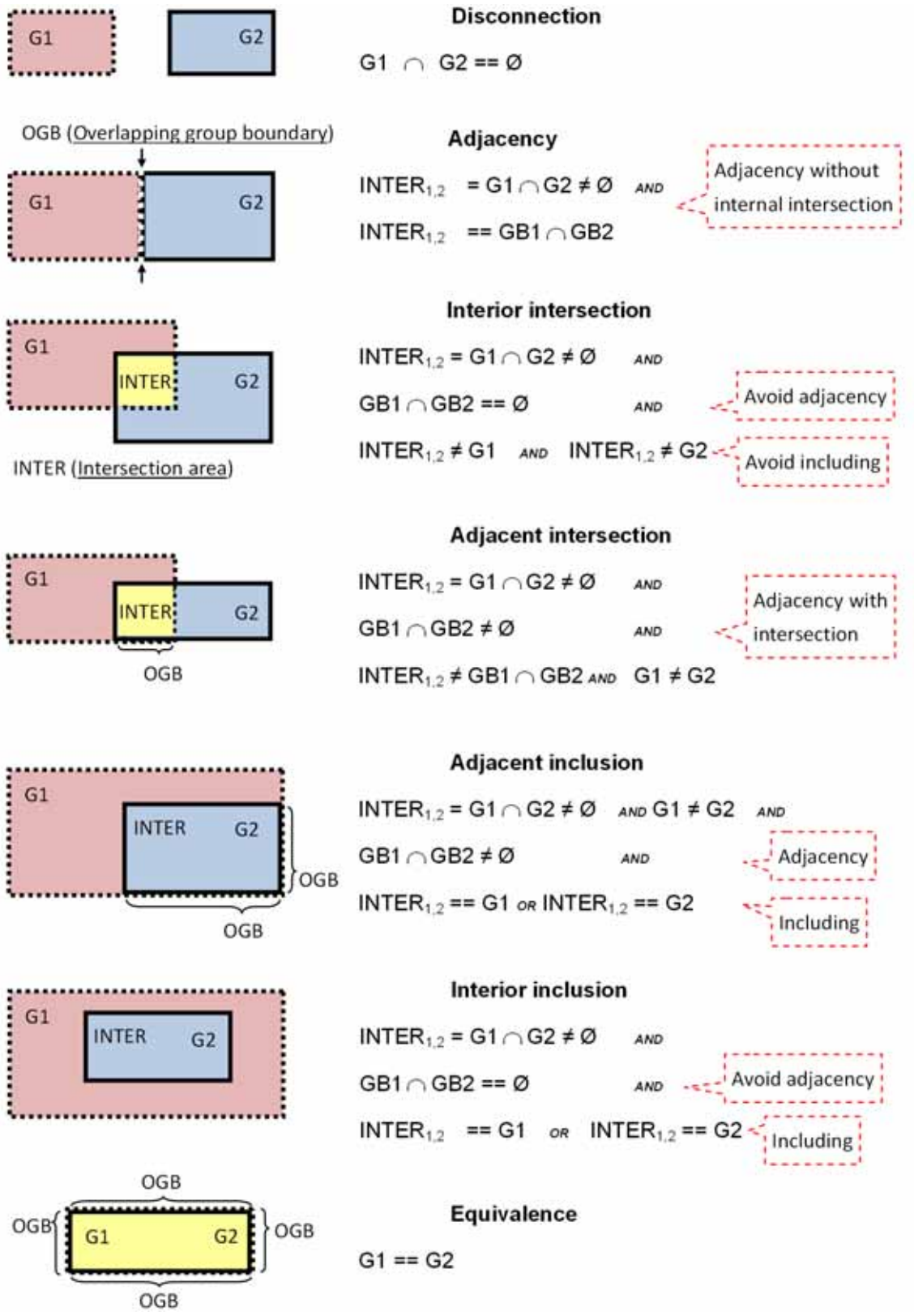


Figure 5.2: Different relative spatial relationships between two groups

- **Inclusion**: the intersection between the two groups concerns the full content of one and only one of the two groups.
  - **Adjacent inclusion**: the intersection part concerns a part of group boundary
  - **Interior inclusion**: the intersection part does not concern at all the group boundary
- **Equivalent**: the intersection part between the two groups coincides with the entirety of both groups

Figure 5.2 shows one example for each case of relationship between two surface groups $G1$ and $G2$ in a 2D mesh model. For each case the corresponding applied condition is indicated on the right side. For sake of clarity, here the groups $G1$ and $G2$ are both simply connected groups (see def.5.1 on page 149). If they are not connected groups, each connected sub-groups can be extracted at first for testing their reciprocal relationships. Both the two groups used in the figure 5.2 contain only mono-dimension elements. In case of two groups of different dimension or two groups of multi-dimension elements we should generate the corresponding mono dimensional groups on which the evaluation is then performed. A mono-dimensional group contains only elements of the same dimension. Therefore a group containing nodes and faces could produce two mono-dimensional groups: a group of nodes and a group of faces. The use of mono-dimensional groups for different treatments is detailed in section 5.2. The abbreviation GB means group boundary, the OGB means overlapping group boundary and the $INTER$ means the intersection area between the two groups.

5.2 Middle-level semantics transfer

The group definition preservation during the FE mesh modification involves two aspects: the boundary of the group extent and the group content (i.e. the associated mesh elements).

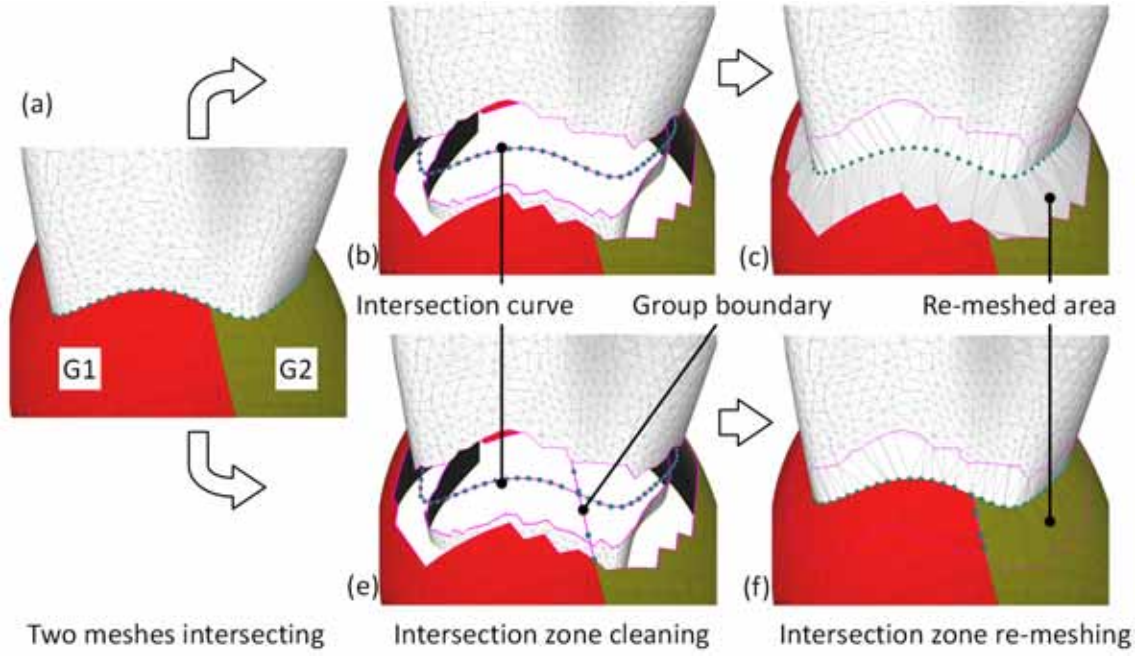


Figure 5.3: Example of group definition preservation during the mesh merging

5.2.1 Preservation of group boundary

The figure 5.3 shows an example of a merge operation of two intersecting meshes on which the triangle groups $G1$ and $G2$ are respectively defined. For merging the two meshes the zone surrounding the intersection curve has to be re-meshed. This example highlights the difference in applying the merging operation by simply considering the mesh geometric and topological information and by using also the group information. In the first case (fig.5.3.b and fig.5.3.c) the re-meshing process takes into account only the intersection curve between the two meshes. The re-meshed area contains newly created mesh elements without any group definition (fig.5.3.c). The direct re-assignment of group data on these newly created mesh elements is not possible as there are some triangles crossing the two groups. To preserve the initial groups during the merging process, the intersection curve as well as the group boundary should be considered while cleaning the intersection zone (removing intersecting triangles) (fig.5.3.e). In this way, the correct re-assignment of group information on the newly created mesh elements during the re-meshing process is possible (fig.5.3.f).

The boundary of a mesh group does not always explicitly exist, because a boundary consists of a set of elements bounding a space of dimension equal to the mesh dimension. But not all the groups represent a space of dimension equal to mesh dimension as, for example, a group of nodes (dimension equal to 0) in a triangular mesh (dimension 2). To handle this problem, the notion of virtual group boundary is proposed and detailed in the following section.

5.2.2 Concepts of Virtual Group Boundary (VGB)

Since the group boundary does not correspond precisely to the topological notion of boundary, it requires a specific definition. This is even more true when the boundary does not exist geometrically (in the case of node group, for example) and should be therefore evaluated. Thus, we introduce a so-called Virtual Group Boundary (**VGB**) as a set of 0D, 1D and/or 2D elements located at the group limit. The dimension of FE entities constituting the group boundary depends on the dimension of the mesh (d_M) as well as on the dimension of the elements constituting the group (d_G). For a given group, the VGB can be decomposed in two potentially empty subsets:

- a set of **Bounding Elements** (**BE**) gathering together the elements of the mesh having a dimension ($d_M - 1$). They may not belong to the group and constitute one or more connected sets enclosing connected portions (in the following referred as domain) of the mesh whose elements of dimension d_G belong to the group,
- a set of **Isolated Elements** (**IE**) belonging to the group and having a dimension d_G.

The reason why we consider these two sets to define the VGB is motivated by the use of their constitutive elements (their topology and position) in the mesh modification process under constraints in order to propagate accurately the group data in the resulting mesh after modification.

In order to compute VGBs of groups in the case of 2D/3D meshes, we have been considering an exhaustive list of possible configurations, as described in the following sections.

5.2.2.1 VGB of groups defined over a 2D mesh

Depending on the dimension of the elements constituting the considered group, three configurations are distinguished and illustrated on figure 5.4:

- The VGB of a group of 2D elements (faces) belonging to a 2D mesh is defined by one set of BE gathering together some of the edges of the mesh. These edges are associated to one and only one face f of the group so that f has at least one edge in common with another face of the group. They define closed loops enclosing one or several connected faces of the 2D group. Figure 5.4.a_1 shows a group of faces whose VGB is identified in figure 5.4.a_2.

- The VGB of a group of 1D elements (edges) belonging to a 2D mesh is defined by two sets of edges. The set of BE gathers together the edges associated to exactly one and only one face f whose edges are in the group, i.e. the other adjacent face of e has at least one edge not belonging to the considered group, and such that f has at least one edge e not considered as BE. They form closed loops that enclose connected mesh areas in which all the edges are components of the 1D group. The set of IE contains the edges associated to no face on which all the edges are in the group or to faces in which all the edges may be classified as BE. In this case, both the BE and IE belong to the group. Figures 5.4.b_1 and 5.4.b_2 show a group of edges and the corresponding VGB made of BE and IE.

- The VGB of a group of 0D elements (nodes) belonging to a 2D mesh is constituted by a set of edges and a set of nodes. The set of BE gathers together the edges associated to one and only one face f whose nodes (all) are in the group, i.e. the other adjacent face of e has at least one node not belonging to the considered group, and such that f has at least one edge e not in BE. The set of IE contains the nodes associated to only faces having none or only one or two nodes in the group or to faces in which all the edges may be classified as BE. This case is illustrated on figures 5.4.c_1 and 5.4.c_2.

In the case of 0D and 1D groups, the elimination of the three bounding edges belonging to the same 2D element (faces) from the set of BE and the transfer

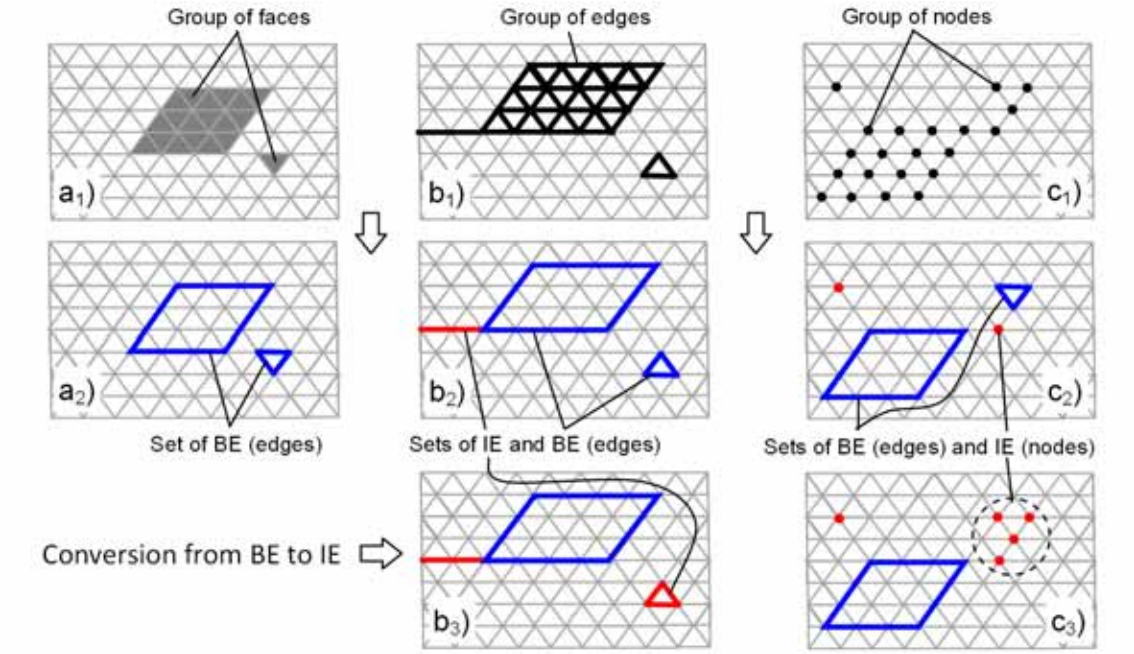


Figure 5.4: Examples of VGBs extracted from groups defined over a 2D triangle mesh

of the related n-D (n equal to the group dimension) elements to the set IE will prevent a further decomposition of the 2D element. Actually, nothing guarantees that new edges or nodes inserted in the concerned face have to be included into the group. On the contrary, according to the proposed VBG computation it results that a group of faces in a 2D mesh does not have any IE faces. This is because in a group G, a face having the same dimension as the triangle mesh is supposed to represent a domain that could be subdivided into more than one faces belonging to the group. Therefore instead of defining the face as an isolated face, three bounding edges are considered. Figures 5.4.b_3 and 5.4.c_3 show separately for the case of 0D and 1D groups the transfer of three BE edges into three IE edges.

A similar classification and definition of VGB can be used in the case of quadrangular meshes.

5.2.2.2 VGB of groups defined over a 3D mesh

Similarly to the case of 2D meshes, depending on the dimension of the elements constituting the group, four configurations can be distinguished for volume FEA

meshes:

- The VGB of a group of 3D elements (e.g. tetrahedral elements) belonging to a 3D mesh is formed by a set of BE corresponding to faces associated to one and only one 3D element of the group. The set of IE is empty.
- The VGB of a group of 2D elements (faces) belonging to a 3D mesh is constituted by two sets of faces. The set of BE gathers together the faces associated to one and only one 3D element for which all the associated faces are in the group and so that not all its faces are classified as BE. The set of IE gathers together the faces of the group associated to no 3D element on which all the associated faces are in the group or to 3D elements for which all the faces could be classified as BE.
- The VGB of a group of 1D elements (edges) belonging to a 3D mesh is constituted by a set, possibly empty, of faces and a set, possibly empty, of edges. The set of BE contains the faces associated to one and only one 3D element on which all the associated edges are in the group and such that not all its bounding faces can be classified as BE. The set of IE gathers together the edges associate to no 3D element on which all the edges are in the group or to 3D elements in which all the associated faces can be classified as BE.
- The VGB of a group of 0D elements (nodes) belonging to a 3D mesh can be constituted by a set, possibly empty, of faces and a set, possibly empty, of nodes. The set of BE gathers together the faces associated to one and only one 3D element for which all the nodes are in the group and such that all its faces cannot be classified as BE. The set of IE contains the nodes relative to no 3D element on which all the nodes are in the group or to 3D elements in which all the associated faces can be classified as BE.

From the above definition, it returns that for group made of tetrahedral elements the BE may also contain all the faces enclosing the same 3D element. This is due to the fact that the associated semantic information is relative to the enclosed volume and it is therefore meaningful to propagate it to new 3D

elements that might be inserted during the mesh modifications. This is not true for all the other group configurations (0D, 1D and 2D groups) in these cases, the elimination of four bounding faces belonging to the same 3D element from the set of BE and the transfer of the related n-D (n equal to the group dimension) elements to the set IE will prevent a further decomposition of the 3D element.

A similar classification and definition of VGB can be used in the case of hexahedral mesh. The VGB computation above is proposed for the mono-dimension groups; therefore a group of mixed dimensions should be divided into mono-dimension groups in order to compute the VGB. In this way each sub mono-dimension group has its proper VGB.

5.2.3 Preservation of group content

Once the virtual group boundary is identified, the mesh modification process should be constrained to preserve this boundary curve to a certain extent. For example, depending on the associated semantics, the mesh deformation presented in chapter 4 ($p_{.85}$) should take into account the group boundary curve as a deformation constraint. In case of re-meshing, the triangulation/tetrahedralisation should be constrained by different elements: modification area boundary, groups' boundary and some other modification interface such as the intersection curve in the case of figure 5.3. Therefore the constrained meshing process should identify and fill correctly each sub-domain that is enclosed by a set of constrained segments. More precisely we identify different closed and not intersecting loops to fill directly.

The first issue to be solved is how to mesh correctly the domain submitted to a modification by taking into account different constraint segments. To see how complex it could be let us consider an example where several groups, some overlapping, are partially in the re-meshing area. Figure 5.5.a presents an example of mesh model that consists in a surface mesh. Different 2D groups are defined on the model and these groups are shown separately in figure 5.5.b. There are five surface groups from $G1$ to $G5$. In this example, the group $G1$ and the group $G2$ are overlapping more precisely $G1$ adjacently includes $G2$. The group $G3$ being disconnected contains two disconnected sub-parts. The two groups $G4$, $G5$ are

also overlapping and more precisely $G5$ adjacently includes $G4$. Let us suppose that the model needs to be re-meshed for achieving a certain mesh modification (for example adapting to another mesh for merging the two). Figure 5.5.c shows that the re-meshing area mesh elements have been removed and the different group boundaries have been kept. The boundary of the empty re-meshing area in which new mesh elements will be created and the different group boundaries constraining the meshing process are shown and zoomed in the figure 5.5.d. In addition, two new triangles are created for meshing: $\Delta 1$ and $\Delta 2$. It is easy to understand that these two newly created triangles do not conform to the initial boundary representation of the model shown in the figure 5.5.a.

The second issue is how to re-assign correctly the group definition on the newly created mesh elements once the re-meshing process terminates. The only information we have is the group information on each group boundary. But by only using the group definition on the group boundary makes complex to assign group information to the newly created mesh elements and the assignment is not guaranteed to be correct. In fact, there are some mesh elements that should be part of several groups bounded by either close or far boundaries. Some group boundary segments have more than one group definition. Some newly created mesh elements should be assigned to groups whose boundary is outside of the re-meshing area.

For resolving the two above issues, constrained meshing and re-assignment of group, we propose to decompose the modification/re-meshing area into several non-overlapping sub-patches which are characterised by the same group set before re-meshing. To this aim we introduce a concept of elementary groups. An elementary group is the sub-set of the group elements associated to the same set of groups, i.e. it is the set of the overlapping parts of the associated groups. Since the re-meshing area can be treated as a special group, elementary groups involving it will correspond to the re-meshing patches bounded by constraint segments corresponding to parts of the boundary of the re-meshing area and/or of the concerned VGBs. Therefore we just need to re-mesh the patch of each elementary group and assign the new mesh elements the concerned groups' definition. Thus for the example given in figure 5.5 the elementary groups should be computed and their boundaries would be those depicted in figure 5.5.e. In this way the

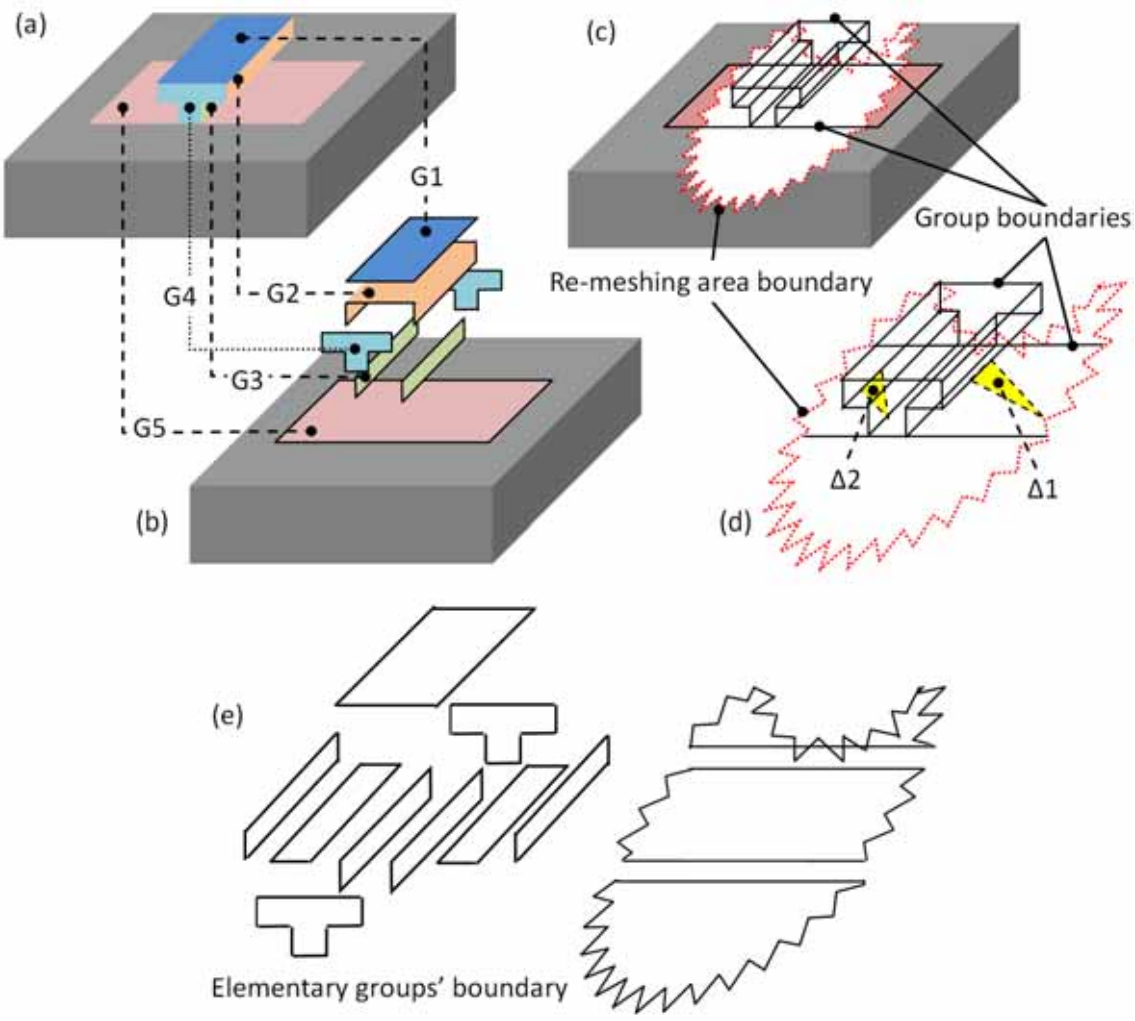


Figure 5.5: Example of needs of re-meshing in the overlapping group area

constrained meshing problem is safely converted into an operation of filling mesh elements in those elementary groups' boundary.

5.2.4 Decomposition into Elementary Groups (EG)

The Elementary Groups (**EG**) are defined as sub-groups containing the FE entities belonging to the same groups of the FE mesh. This concept appears at group representation level (or middle level between the semantic and the geometric ones, see section 3.1 on page 69). Thus, the subdivision into EGs will

neither affect the geometric entities nor the semantic data. Actually, EGs are created from original groups (FE groups, recognised surface patches, re-meshing zone etc.) to avoid partially overlapping configurations except for elements on the VGB, to ease the constrained re-meshing. To sum up, between two EGs the relationships acceptable are only disconnection and adjacency. If two groups are spatially overlapping the EGs decomposition can be performed. The developed EGs decomposition can be applied to groups of different dimensionality such as a group of nodes and a group of faces, including those of mixed-dimensionality, e.g. a group containing nodes and faces. In this case the group is first decomposed into the appropriate mono-dimensional groups.

The data structure to handle the EGs is the array illustrated in algorithm 10. The integer *dim* indicates the dimension of the elements contained in the EG. The list *domain* gathers together all the elements of dimension d_M (mesh dimension) covered by the entities of the EG. For example, when an EG of a 2D mesh contains a set of nodes, the list *domain* gathers together the underlying faces to which the nodes belong. When an EG of a 3D mesh contains a closed set of faces, the list *domain* gathers together the 3D elements bounded by this set of faces. The two lists *bounding* and *isolated* respectively contain the BE and IE of the VGB. The list *groups* includes the groups from which this EG was created. The list *entities* corresponds to those mesh elements forming the EG extracted from the original groups. Actually, BEs enclose the space of dimension d_M defined by elements included in the list *domain*. IEs can be considered as punctual elements, which cannot form a space of dimension d_M.

Algorithm 10 Data strucutre of Elementary Group

```
Structure ElementaryGroup
  dim       As Integer // 0: node, 1: edge, 2: face, 3: tetra or -1: mixed
  domain    As List    // mesh elements of d_M for representing group space
  bounding  As List    // VGB's bounding elements (BE) of dimension d_m − 1
  isolated  As List    // VGB's isolated elements (IE) of dimension dim
  groups    As List    // associated FE groups, re-meshing area groups, etc.
  entities  As List    // mesh entities in this elementary group
End Structure
```

To summarize the developed algorithm for creating EGs from different input

Algorithm 11 EGs splitting

```
Function elementaryGroupSplitting (EG1 As List, EG2 As List) As List
 1: Variable newEGs As List
 2: Variable newEG1, newEG2 As ElementaryGroup
 3: Variable interDomain, subDomain1, subDomain2 As List
 4:
 5: if EG1.domain ≠ EG1.domain then
 6:   interDomain ← EG1.domain ∩ EG2.domain
 7:   if interDomain ≠ φ then
 8:     subDomain1 ← EG1.domain − interDomain
 9:     subDomain2 ← EG2.domain − interDomain
10:
11:     if subDomain1 ≠ φ then
12:       addElementToList(newEG1, newEGs)
13:       for all e in EG1.entities do
14:         if e associates with elements in interDomain then
15:           addElementToList(e, newEG1.domain)
16:           if e does not associate with any elements in EG1.domain then
17:             removeElementFromList(e, EG1.entities)
18:           end if
19:         end if
20:       end for
21:     end if
22:
23:     if subDomain2 ≠ φ then
24:       addElementToList(newEG2, newEGs)
25:       for all e in EG2.entities do
26:         if e associates with elements in interDomain then
27:           addElementToList(e, newEG2.domain)
28:           if e does not associate with any elements in EG2.domain then
29:             removeElementFromList(e, EG2.entities)
30:           end if
31:         end if
32:       end for
33:     end if
34:
35:     for EG1, EG2, newEG1 and newEG2 do
36:       update the fiels domain, bounding, groups, elem_dim
37:     end for
38:   end if
39: end if
40: return newEGs
End Function
```

groups (EGs, FE groups, sub-meshes corresponding to recognised surface types, re-meshing area group, etc.) consists of the following main steps:

1. each group is transformed into one or more (for mixed dimensional groups) EGs containing elements of the same dimension, so that the field *entities* contains all the elements of the group;

2. for each EG the field *domain* is updated by considering all the underlying mD elements having all their associated nD elements belonging to this EG. The mD is the dimension of the considered FE mesh and the nD is the dimension of the elements in EG. The *bounding* field is updated by looking for all the $m-1$ D elements associated to exactly only one mD element in the *domain*. The *isolated* field gathers together all the elements contained in the field *entities* which are not associated to elements of the *domain*;

3. set operations are performed between all the couples of EGs to potentially produce new non partially overlapping EGs. These computations use the algebra of 2D and 3D sets to evaluate the relationships between the initial EGs. When partially overlapping configurations are detected, the initial EGs are split. The algorithm 11 is used to produce those EGs from two potentially partially overlapping groups EG_1 and EG_2.

Figure 5.6 illustrates the complete algorithm on an example of two partially overlapping FE groups. The process starts from two Elementary Groups EG_1 and EG_2 (fig.5.6.a) directly computed from two FE groups of dimensions 2 (group of faces) and 0 (group of nodes) respectively. Each EG is then treated to identify the mesh elements representing the group *domain*, its *boundary* and the *isolated* elements (fig.5.6.b_i and fig.5.6.c_i). The intersections of the two initial *domains* are then computed (fig.5.6.d_2) to enable the definition of four non-partially overlapping EGs (fig.5.6.e_i).

To illustrate the results of the EGs computation from an initial set of FE groups, we use an industrial example of figure 5.7 corresponding to a triangular mesh model of a part of a caisson involved in a fast operational study of EDF. At the geometric level, the mesh is defined by 38672 nodes, 115253 edges and 76582

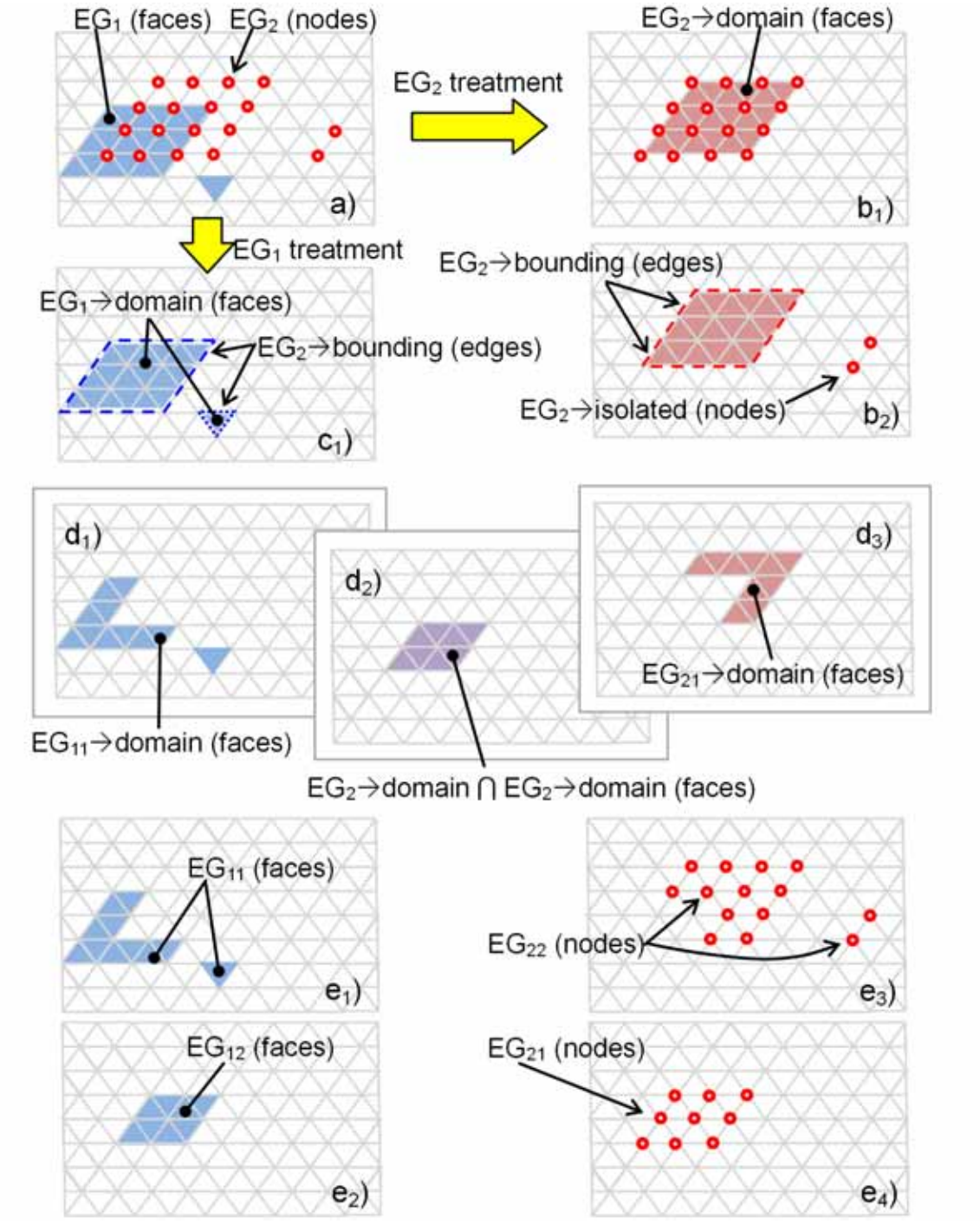


Figure 5.6: Definition of EGs from two partially overlapping groups of nodes and faces

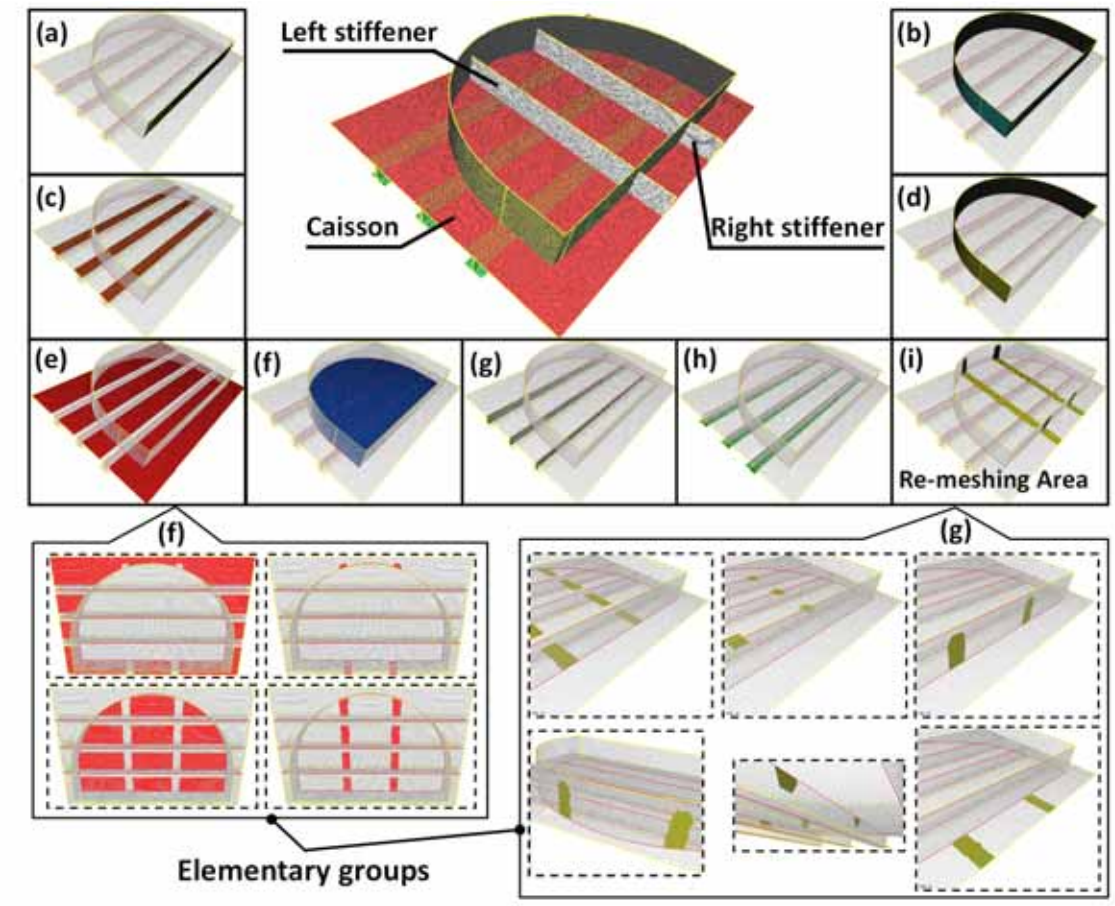


Figure 5.7: EGs computation on a triangular mesh model of the caisson (courtesy EDF R&D)

triangles. At the group level, the model is structured with 8 groups (fig.5.7.a - fig.5.7.h) of faces that partially overlap. Two stiffeners (left, right) have to be merged with a part of the caisson to strength the structure. Therefore, the re-meshing area is computed on the caisson and defined as a group (fig.5.7.i).

The elementary group decomposition algorithm is performed on this model and produces in total 28 EGs. For example, the FE group shown in the figure 5.7.e is decomposed into four EGs (fig.5.7.f). The re-meshing area is also decomposed into six EGs (fig.5.7.g) according to the interacting FE groups.

5.3 High-level semantics transfer

During the mesh modification process the associated physical semantics, required for FE simulation, has to be preserved as discussed in the previous sub-section. The preservation of semantics is done mainly on the middle level semantics through the group definition. Different characteristics of groups have been analysed and how to preserve these characteristics is presented previously.

This sub-section focuses on the high-level semantics aspect. After performing model modifications the FE mesh groups are preserved without looking at the associated semantics, as introduced previously. Here the semantics is also analysed in order to check if it is still "compatible" with associated mesh elements after modifications.

The transfer of the high-level semantics requires analysing the meaning of the semantics to transfer the semantics while mesh modification. The transfer concerns the semantics location (i.e. supporting groups) as well as the associated numerical value (e.g. physical parameters required for FE simulation, shape parameters, $\cdots$). The **location** has a spatial characteristic and specifies which part of the model is involved for its specification. Since the high-level semantics is associated to the geometry model through the intermediate group definition therefore the location of the semantics concerns the mesh elements contained in the relative groups. The **value** concerns the content of semantics such as quantity of pressure, direction and value of a nodal force or displacement, shape category of a surface as well as the parameters (e.g. radius, axis, $\cdots$), etc.

First of all we tried to summarise the possible high-level semantics and to list them according to the supporting group dimension. Then several proposals of semantics transformation/propagation due to mesh modification are presented.

5.3.1 Proposal of high-level semantics classification

In order to deal with the high-level semantic data transfer, we are taking into account the following characteristics:

- supporting group dimension (section 5.1.1 on page 148),

- nature of the type of the semantic information such as FEA (displacement/force BCs, geometric/mechanical characteristics), model shape (plane, sphere, circle, $\cdots$), etc.
- objective of the group creation in case of FEA (numerical modelling, post-processing and analysis of results), or in case of constrained shape modification (re-meshing area, constrained shape deformations), etc.

From the possible different Finite Element Analysis (FEA) semantics, the most used by EDF are here summarised according to the dimension of the groups they are associated to:

- 0D groups (nodes):
  - Type of mechanical modelling (idealisation): discrete elements as springs, punctual mass
  - Mechanical characteristics of discrete elements: stiffness, mass
  - Displacement BCs: fixing conditions, displacements following a given direction, link conditions between disconnected parts of the structure, contact conditions
  - Force BCs: forces/moments following a given direction/axis
- 1D groups (edges):
  - Type of mechanical modelling (idealisation): beams, pipes
  - Geometric characteristics of beams: section properties
  - Mechanical characteristics of beams: material characteristics like Young's modulus, Poisson's ratio, non-linear behaviour laws
  - Displacement BCs: fixing conditions, displacements/rotations following a given direction, link conditions between disconnected parts of the structure
  - Force BCs: forces/moments following a given direction/axis, pressure
  - Mechanical state of the structure: initial stress/deformation

- 2D groups (faces):
  - Type of mechanical modelling (idealisation): plates, shells, interaction surface fluid-solid, etc.
  - Geometric characteristics of shells: thickness, position of the middle surface
  - Mechanical characteristics of shells: stiffness, material characteristics like Young's modulus, Poisson's ratio, non-linear behaviour laws
  - Displacement BCs: fixing conditions, displacements/rotations following a given direction, link conditions between disconnected parts of the structure, contact conditions
  - Force BCs: forces/moments following a given direction/axe, pressure
  - Mechanical state of the structure: initial stress/deformation
- 3D groups (volume):
  - Type of mechanical modelling: 3D
  - Mechanical characteristics: material characteristics like Young's modulus, Poisson's ratio, non-linear behaviour laws
  - Displacement BCs: fixing conditions, linear displacements following a given direction, link conditions between disconnected parts, etc.
  - Mechanical state of the structure: initial stress/deformation

Out of FEA context, the shape semantics is considered and classified as follows:

- 0D groups (nodes): punctual shape which consists of Cartesian coordinates (x,y,z), this punctual shape could be defined by intersection between two curve shapes defined by 1D groups.
- 1D groups (edges): curve shape which is defined by several edges or points. The straight line (linear curve) is defined by two nodes which correspond to the segment extremes. The non-linear curves are defined by more than 2 nodes. For example a circle (quadratic curve) is defined by at least 3 points.

- 2D groups (faces): surface shape which is defined by several points or several curves. A plane could be defined by 3 non-aligned points or 2 non-parallel straight lines. Non-linear surfaces such as sphere, cylinder, etc could be also supported by face groups. There are certainly other semantics with other objectives that could be defined on groups of different dimensions. We could not list all possible semantics information but we want to show the main idea through the FEA semantics and shape semantics.

5.3.2 Towards rules for high-level semantics transfer

As stating in the beginning of this section, during the mesh modification the high-level semantics might need to be transferred (updated, adapted, propagated, etc.) in order to keep compatible with the modified geometry model. We try to summarise the different possible geometrical changes relative to group elements:

1. **Group neighbourhood addition**: new mesh elements are created and connected to mesh elements belonging to a group supporting a specific semantics. For example applying a material addition operation new tetrahedra are created and attached onto the original tetrahedra which are belonging to a semantics group

2. **Group neighbourhood deletion**: when mesh elements adjacent to other elements in a group are deleted. For example when applying a material deletion operation, neighbour tetrahedra adjacent to a semantics group tetrahedra are deleted.

3. **Group neighbourhood affine transformation**: when mesh elements adjacent to other elements in a group are shifted, rotated, scaled and sheared. In this case there may not be any affectations to the concerned group.

4. **Group element addition**: when new mesh elements are created and added in a group, it is enough to assign these new elements the concerned group definition. For example, in case a mesh refinement are applied on the mesh elements belonging to a group in this case newly added mesh elements should also associate with the concerned group

5. **Group element deletion**: mesh elements belonging to a group associated with a specific semantics are deleted. This is the case of material deletion operations such as cutting, which produce the removal of mesh elements.

6. **Group affine transformation**: Mesh elements gathered by a semantics group are shifted, rotated, scaled and sheared. For example nodes in a semantics group could change their position; nodes that support the triangles in a semantics group are moved from a plane onto a sphere so that the surface represented by these triangles is transformed from a plane into a sphere.

For each of the above cases, a simple example of geometric modification is shown in figure 5.8. Figure 5.8.a presents a model on which a surface group SG is defined. This model is then modified by applying a material removal (fig.5.8.b) that leads to both a group elements and a group neighbourhood deletion. The half cylindrical hole feature insertion removes the part of surface of the group SG. An addition operation on the same model brings new neighbours to the existing group SG (fig.5.8.c). An affine transformation is also performed on some group elements of the same model (fig.5.8.d). The surface of the group SG is changed. The deletion of group elements and of group neighbourhood can happen also separately, as in the case of figures 5.8.e, 5.8.f and 5.8.g. In this example the initial model has three volume groups $G1$, $G2$ and $G3$, a material removal operation is first performed on the part without any group associated (fig.5.8.f) and then it is also performed on a part of the group $G1$ (fig.5.8.g).

The addition and deletion of group neighbourhood are very important because they will change the property of the groups in terms of what the groups "touch". In case of neighbourhood deletion the group elements will change from "touching" some geometry elements not in a group or in a different group towards "touching" nothing. The neighbourhood addition will make the changes on the groups in an opposite way. Similarly the deletion of group elements will make some resting group elements changing from "touching" the group elements to "touching" nothing. With these changes certain attached semantics may results invalid or may have to change. The group affine transformation usually changes the semantics relative to the position, direction and shape.

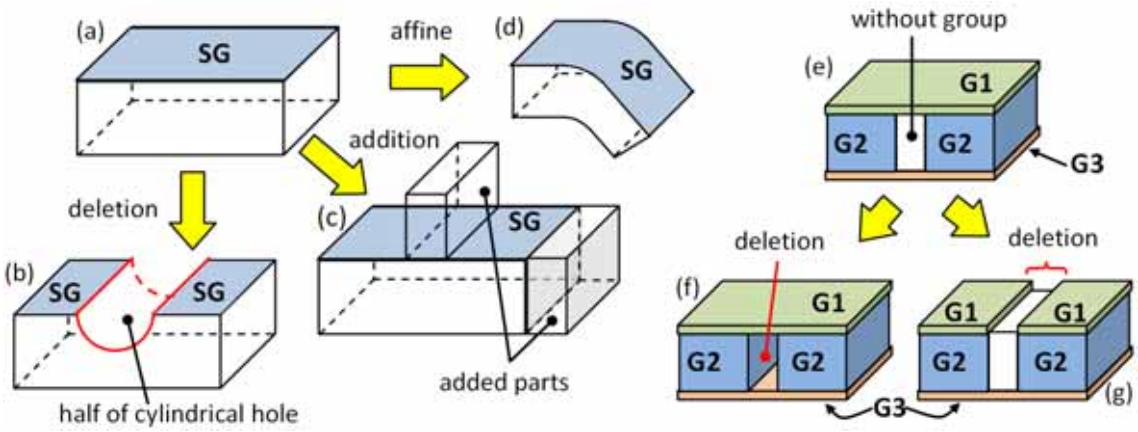


Figure 5.8: Geometric modification around groups

Once the four possible geometric changes around groups are identified let us discuss how to react in terms of semantics. In this thesis the high-level semantics transfer remains still at a proposal level, so that we try to use some examples to show how this proposal should be developed further. Therefore, for each of the identified four possible changes we consider an example with different semantics listed in section 5.3.1.

Let's consider the example shown in the figure 5.9 which represents an object containing a fluid (fig.5.9.a). The half of the corresponding numerical simulation model is shown in figure 5.9.b. The model has an internal part which has a material B and an external part which has a material A. Since there is fluid the top facet of the internal part with material B as well as the internal wall of the part with material A receive a fluid pressure (red arrow). Two structural modifications are considered in this example: making a half hole on the top facet of the part with material B and adding a stiffener on the higher part of the hull with material A.

1. The half hole made on the top facet of the internal part with material B is shown in the figure 5.9.c. Consequently the groups supporting the two semantics (material and fluid pressure) the material B need to be updated. First the volume mesh elements that are deleted from the mesh due to the half hole insertion should be removed from original group associated to the material B and the fluid pressure is not defined anymore on the disappearing

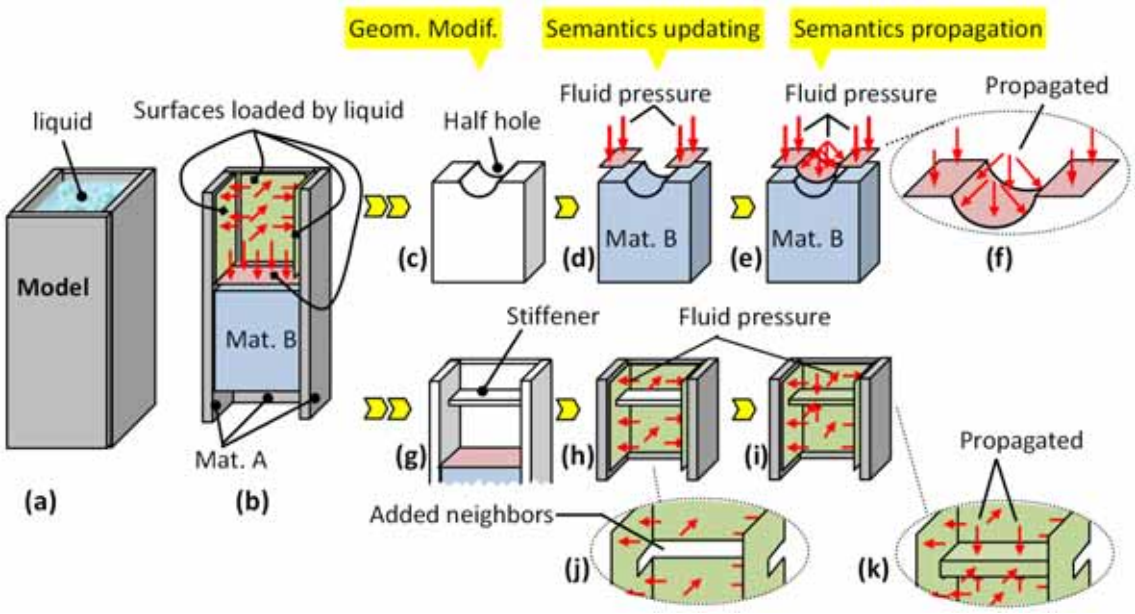


Figure 5.9: Example high-level semantics transfer

surface geometry elements (fig.5.9.d). It is easy to understand that in case of *Deletion* of semantics associated mesh elements, the link between the semantics and mesh elements to be removed should be broken down.

Actually the operation performed includes also the case of *Group neighbourhood deletion*; in fact, the geometry elements whose neighbours associated with semantics are removed and then additional checks have to be performed to check if additional operations have to be performed at the semantics-intermediate information level.

In this specific case, a complete semantics propagation requires some additional changes: in the reality the fluid can fill in the half hole made and the new appearing surface part touches the fluid. The top surface geometry elements actually have lost neighbour volume geometry elements so, due to the specific semantics (fluid pressure), these surface geometry elements can inherit the property of the removed top surface elements: in touch with liquid. Thus the semantics propagation should include the half hole wall between the two original top sub-facet (fig.5.9.e and 5.9.f) in the area subjected to the fluid pressure semantics.

2. The addition of the stiffener on the model external part is shown in the figure 5.9.g. In this case the *Group neighbourhood addition* is involved, these mesh elements having new neighbours do not associate with the semantics fluid pressure anymore. This change cannot be applied in all cases it depends strongly on the attached semantics. So in this case, the volume affected by the material B definition parties not modified, thus only the fluid pressure defined on the external part wall will be changed. Because the surface part where the stiffener is merged doesn't touch anymore the liquid, but the material of the stiffener the pressure should not be anymore defined on these surface geometry elements (fig.5.9.h and 5.9.j). In the zoomed picture (fig.5.9.j) only the original wall of the external part is shown and the zone where the stiffener is glued is colored in white. The green surface receives the fluid pressure as before.

   Moreover it has to be noted that the added stiffener is partially exposed to the fluid, i.e. except for the elements directly in contact with the original wall elements. Therefore it is also very reasonable to propagate the fluid pressure onto the stiffener hull surfaces which is exposed to fluid. Always in the cases like this of *Group neighbourhood addition* the new neighbours of the geometry elements associated with the semantics may be affected by this semantics.

In the above geometry modifications the topology of the model is not changed. In the first modification, the top facet of the part of material B is changed from completely planar to partially cylindrical, while in the second modification the shape of the wall is changed through a kind of extrusion. These two modifications are *Group affine transformation* for the fluid pressure associated surface therefore the associated data values (vectorial, scalar) might be changed. In this example the pressure should be always perpendicular with the touched face.

From an implementation point of view, the semantics propagation could be done in an automatic or in a supervised way. We think that the latest decision about group and values modifications has to be made by the expert that can decide to accept or not the proposed propagation.

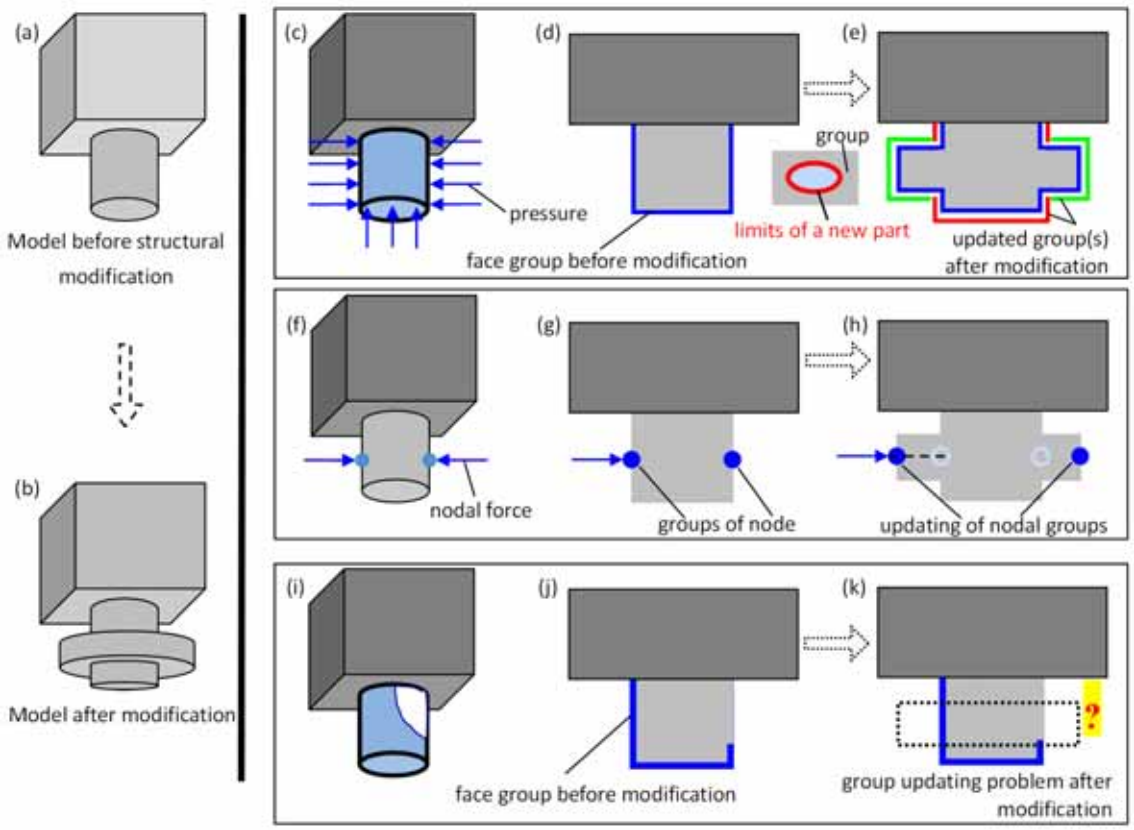


Figure 5.10: Examples of possible high-level semantics transfer

As previously said, here we do not provide a complete solution to the transfer problem, but we simply illustrates some aspects useful to identify some possible rules highlighting their strongly dependency on the semantics to be transferred. Figure 5.10 shows possible solutions for face/nodal group updating during the structural modification operation of material adding (fig.5.10.a and 5.10.b).

Figure 5.10.c shows the presence of a group made of boundary faces of the lower sub-part defining the surface pressure; the updating operation for this group can be differently accomplished: simply updating the group by eliminating the elements covered by the added volume (red solution), or new external 2D elements can be used to apply the pressure after the structural modification, they can be either added into a new mesh group (green solution) or added to the existing group (blue solution). In our opinion the best solution for this case is that the system identifies the three groups “blue”, “red” and “green” (fig.5.10.e), and then

the user decides which one to use.

In general, additional criteria can be required to check the possibility to apply the rule of (semi-) automatic proposition of new groups. Figuref 5.10.i, 5.10.j and 5.10.k illustrate an example in which it is difficult to propose a solution to update the blue group because the face elements contained in this group do not totally cover the added sub-part (fig.5.10.i and fig.5.10.j). After mesh modification (fig.5.10.k), it is not possible to propose automatically the updating group solution, and we let the user do it manually. Figure 5.10.f shows an example where boundary node on the original model before modification forms a group for the definition of a nodal force. For this specific semantics, a possible rule could involve the projection of the node group on the new boundary nodes on the added part (according to the punctual force direction specified by the user, for example).

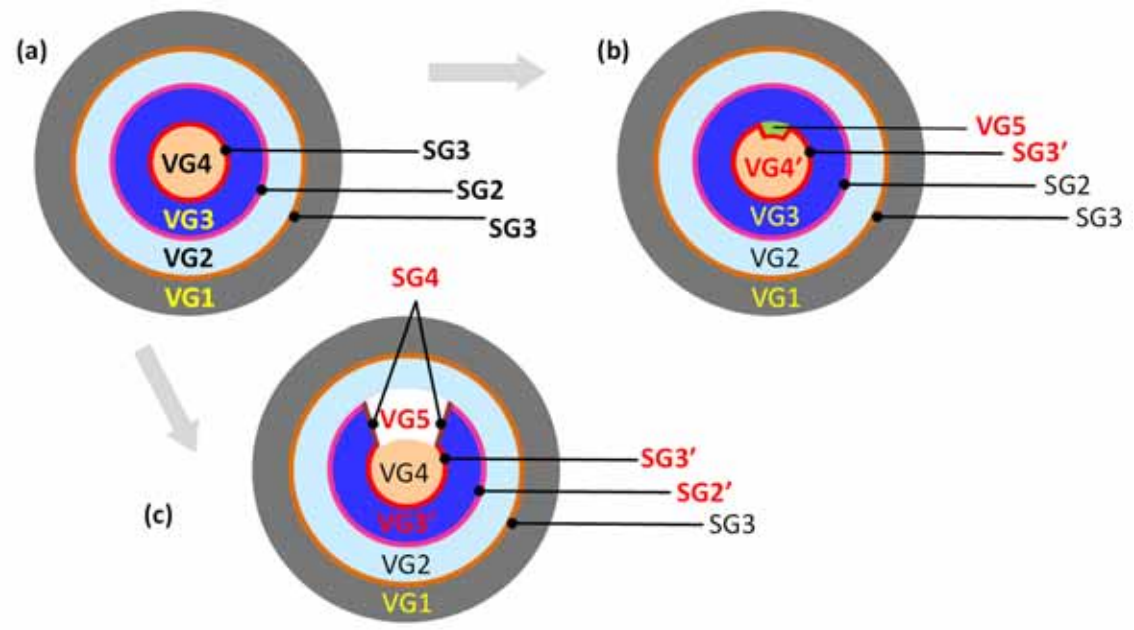


Figure 5.11: Examples of possible high-level semantics transfer

Figure 5.11 shows an example in which the topology of the group part is modified. Figure 5.11.a presents an initial model in which there are 4 volume element groups. $VG1$ and $VG3$ have different materials of solid; $VG2$ and $VG4$ have different fluid. Three surface element groups were created at contact zone

with the fluid: $SG1$, $SG2$ and $SG3$.

The first modification is shown in figure 5.11.b. A new part (green) is added in the material of group $VG4$ at boundary zone. In this case $VG4$ is updated by removing disappearing material and new volume group $VG5$ is created. At surface group level the group $SG3$ is updated by adding the new faces between the material $VG4$ and $VG5$.

The second modification is shown in figure 5.11.c in which a part of material of $VG3$ is removed at the white zone. In this case the volume group VG3 should be updated by removing volume element then the two existing surface groups $SG3$ and $SG2$ should be updated by removing disappearing faces. Since material of $VG2$ and $VG4$ is fluid so the new empty space due to material removing will be occupied by material of $VG2$ and $VG4$. If $VG2$ and $VG4$ concern the same liquid either of both could be defined on this new volume otherwise a new group representing mixed liquid. So we should also created a new volume group $VG5$ for this newly occupied space, and also new surface group $SG4$ for separating the new volume group and existing solid volume group.

In summary, the semantics transfer could be performed at different levels of automation: updating and propagation, in the first case the original group is preserved as much as possible by simply considering those representing the unmodified shape, in the second case the supporting group is modified by including also the changed/added part elements.

Even if general rules always applicable cannot be provided we can identify some trends in the conditions and rules of each in case of different changes previously listed.

In case of group elements *deletion* the semantics should be updated by removing the association with the concerned deleted elements, i.e. removing the elements from the concerned group. If the deleted elements were connected with other elements not associated with the concerned semantics these neighbour elements should be taken into account. This corresponds to the case *Neighbourhood deletion*. According to different semantics it could be propagated onto these neighbours or not.

In case of *Neighbourhood addition* onto group elements, depending on to the semantics the affected elements can remain or not in the original group. Similarly,

depending on the semantics, some of the adjacent newly added elements could be characterised by that specific semantics, i.e. included in the above group.

In case of deforming geometrically the model by applying an *Affine transformation* according to different semantics the associated data values may need to be updated.

5.4 Conclusion

In this chapter, different basic elements and methods for semantics transfer are presented mainly concentrating on from the middle-level semantics layer while also showing the requirements and impact on the high-level semantics.

In our approach the middle layer semantics expressed through the group concept is the key mean for the FEA (high level) semantics transfer. Therefore, the appropriate transfer and updating of the group elements during the FE mesh modification process is extremely important. This is done by preserving the boundary and domain of groups. The notions of **Virtual Group Boundary** and **domain** have been introduced in order to compute the key characteristics of the group independently of the group dimensions. Then the notion of the elementary groups has been proposed to correctly update the group definition after mesh modifications requiring re-meshing, i.e. removal and creation of new elements, of some areas. The virtual group boundary proposed at present encloses a domain (area) which has a dimension equivalent to the dimension of the mesh. This may be not enough because there are groups that don't occupy a space of dimension equal to the mesh dimension. For this reason the virtual boundary of a group is extended to cover more situations. For instance, these elements are considered as isolated so that they can be used as constraints during the mesh modification process. The possible solution is to consider the domain in all dimensions such as a curve domain (i.e. space occupied by a set of edges), the surface domain (i.e. space occupied by a set of faces) in a volume mesh etc.

For transferring the high-level semantics during the mesh modification, we have divided the process into two stages: updating and propagation. We also have summarised four possible geometry modifications in which the mesh elements might be involved to analyse their impact on the high-level semantics

updating and propagation. Then rules for high-level semantics transfer according to different mesh modification categories are introduced. These rules should be further improved in order to guarantee always valid results according to the various shape configurations and types of semantics.

Chapter 6

A generic CAD-less mesh modelling operator and its instantiations

The generic CAD-less mesh modelling approach for modifying semantically enriched meshes has been introduced in chapter 3 (p.69). The generalised operator (fig.3.7 on page 83) and its different components have also been introduced in the same chapter. In chapter 4 (p.85), the different basic methods and tools for manipulating the meshes have been presented. Different aspects for treating the semantics have been detailed in chapter 5 (p.147). In this section, several instances of the generic CAD-less mesh modelling operator are created to answer the needs in mechanical engineering. The first operator is **mesh merging** for gluing two intersecting FE triangular meshes. The second operator is **mesh cracking** that inserts a planar crack feature into a triangular/volume mesh. The third operator is **mesh drilling** that makes "cylindrical hole" on a triangular/volume mesh. The last operator to be presented is **mesh filleting** that will round sharp features on meshes. All of them are based on the same generic CAD-less mesh modelling operator introduced in chapter 3.

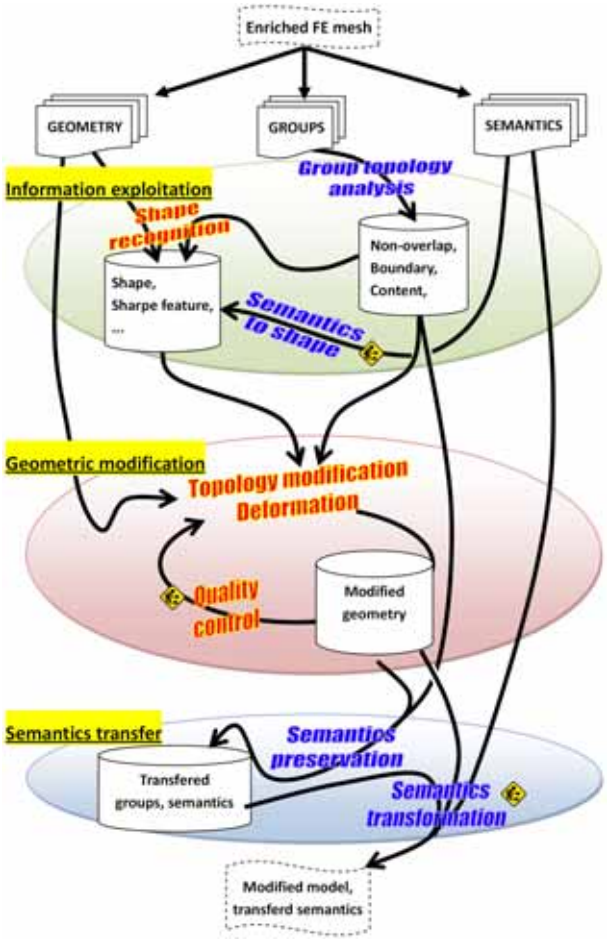


Figure 6.1: Component-based of the CAD-less mesh modelling approach (fig.3.7 on page 83)

6.1 Structure of the operator (recall)

To take into account the multi-layered information during the FEA mesh modification, a generic CAD-less mesh modelling operator concept has been introduced following a component-based approach that contains computation steps removable and re-arrangeable. This generic CAD-less mesh modelling operator designed in this way allows reusing existing basic operators acting on the different information layers and adding new ones. As it will be shown in this chapter, such a framework eases the definition of specific enriched FE mesh modification tools taking into account different geometric and semantics requirements. The CAD-

less operator's general structure with its procedure schema is recalled on figure 3.7. The operator takes as input a semantically enriched mesh for modifying it while transferring the already present semantics. The operator is consisting of three important stages: information exploitation, geometric modification and semantics transfer. The geometric, group and semantic information are exploited at first. These information datas are used to constrain the geometry (mesh) modification. The semantic and group information are updated at the end. The different processes are categorized into geometric (red text) and semantic (blue text) processes.

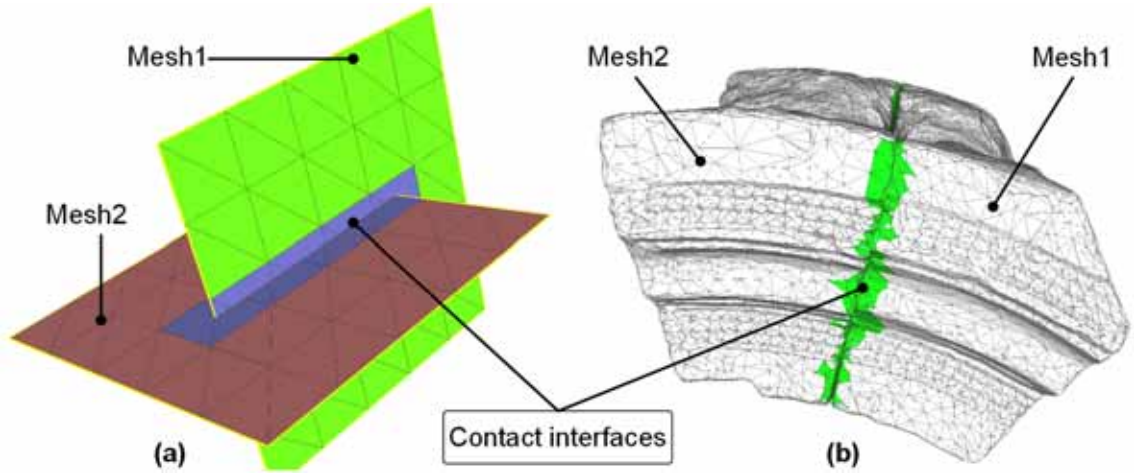


Figure 6.2: Two contact modes between two triangle meshes: (a) face/edge, (b) face/face

6.2 Mesh merging operator

In this section the proposed operator for merging two semantically enriched triangle meshes is presented [75, 76]. Two triangular meshes could be in contact in two ways: (1) face by edge and (2) face by face. An example of each way is shown in the figure 6.2. Two triangular meshes will intersect according to the face by edge mode when the intersection interface is made of triangles that intersect sufficiently. By contrast, two meshes will be in contact according to the face by face mode when the intersection points do not exist or they do exist but do not form a

continuous intersection curve. Anyhow, today, the type of contact is still decided by the user. The algorithms presented in this section enable the merging of two triangle meshes in the face by edge mode, so that the resulting mesh supports the semantic information of the two initial ones. At the end, the face/face mode merging is also presented by using a part of the face/edge merging process.

The overall process of the mesh merging is shown at first. From the identification of the intersections to the re-meshing process, the groups definitions are always taken into account. Then different steps are detailed after. In the rest of this subsection, Mesh1 and Mesh2 represent the two meshes that have to be merged.

6.2.1 Overview of the enriched meshes merging process

To be able to treat the face/edge merging mode, a generic approach is proposed and illustrated on an academic example representing the intersection between a half-vase and a half-cylinder. Two groups of faces are defined on the half-vase (fig.6.3.a). The details of the algorithm are given in the next section:

1. the contact interfaces are firstly identified while looking for couples of intersecting faces. The intersection points are then computed and gathered together in a set of intersection curves (fig.6.3.b);

2. the intersection curves are then optimised so that the number of nodes as well as their position takes into account the size of the surrounding meshes (fig.6.3.c);

3. the contact zone is defined around the intersection curve more precisely in n^{th} neighbourhood which depends on the size of the surrounding triangles;

4. the contact zone is defined as a group and all possible Elementary Groups (**EG**) are computed with the other existing groups by using the EG computing method presented in chapter 5 (p.147). Then the Virtual Group Boundaries (**VGB**) for different EGs are computed;

5. the different EGs belonging to the contact zone are cleaned (fig.6.3.d_1). The VGBs of different groups as well as the boundaries of the meshes are

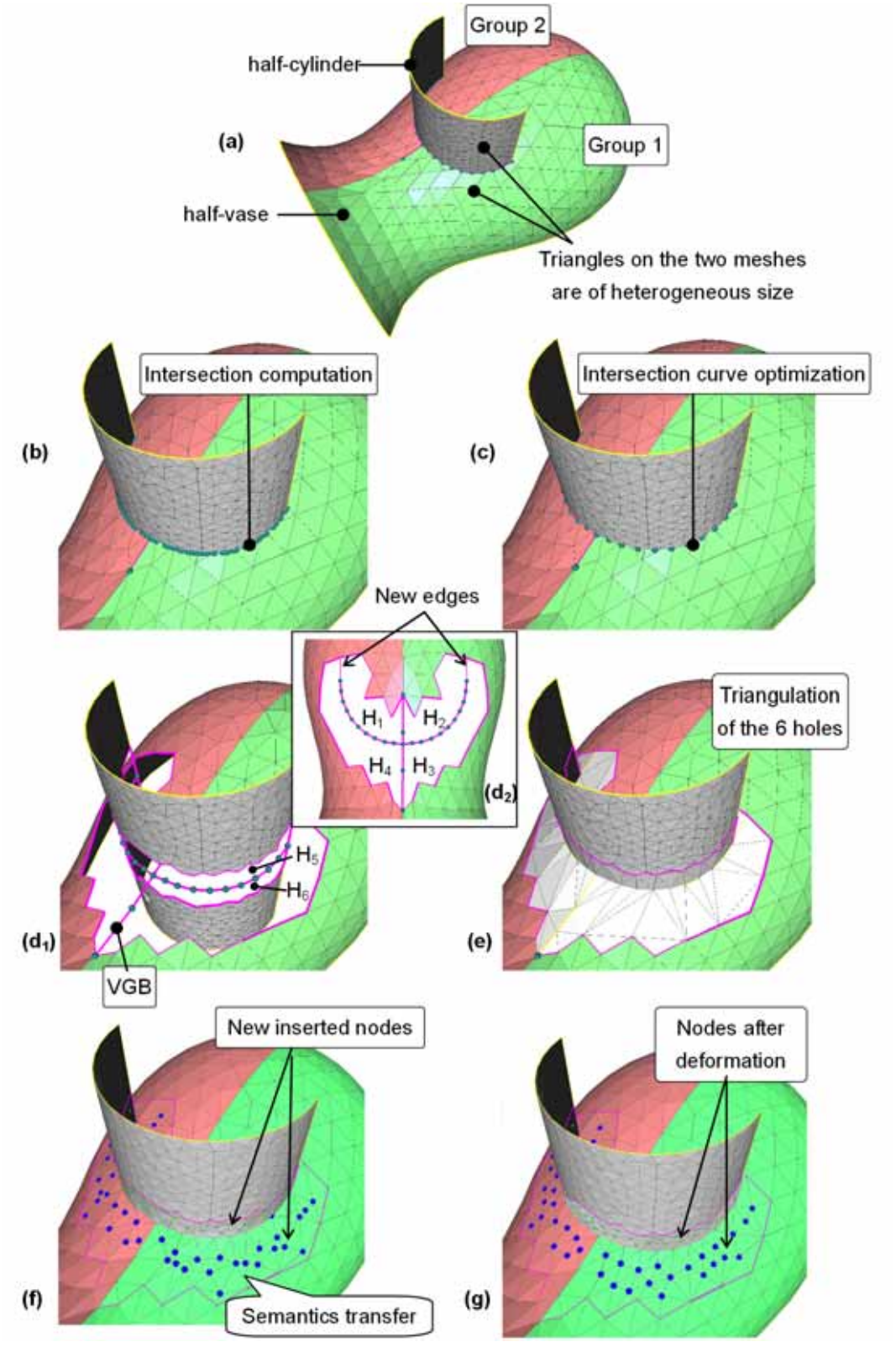


Figure 6.3: Overview on the merging algorithm for two enriched meshes

preserved. The boundaries of the groups as well as the boundaries of the meshes are preserved. The opened intersection curves are closed by newly inserted edges (fig.6.3.d_2). The different elementary patches become a set of holes that now have to be filled by using information coming from the VGBs and EGs identified in the preliminary phase (6 holes H_i in the present case);

6. all the holes are then triangulated (fig.6.3.e) and the semantics defined on the original mesh elements are assigned on the newly created mesh elements (fig.6.3.f);

7. new nodes are then inserted to create a smooth evolution of the triangles' size (fig.6.3.f);

8. the re-meshed zones are relaxed by using the deformation tool to improve the aspect ratio of the mesh elements (fig.6.3.g);

9. the the semantics propagation should be performed from one merged part to another. This is not yet implemented.

The proposed algorithm does not require any action from the user (except potentially step 9). All the parameters are computed automatically. Therefore, on a user point of view, all the steps between the figures 6.3.a and 6.3.g are hidden.

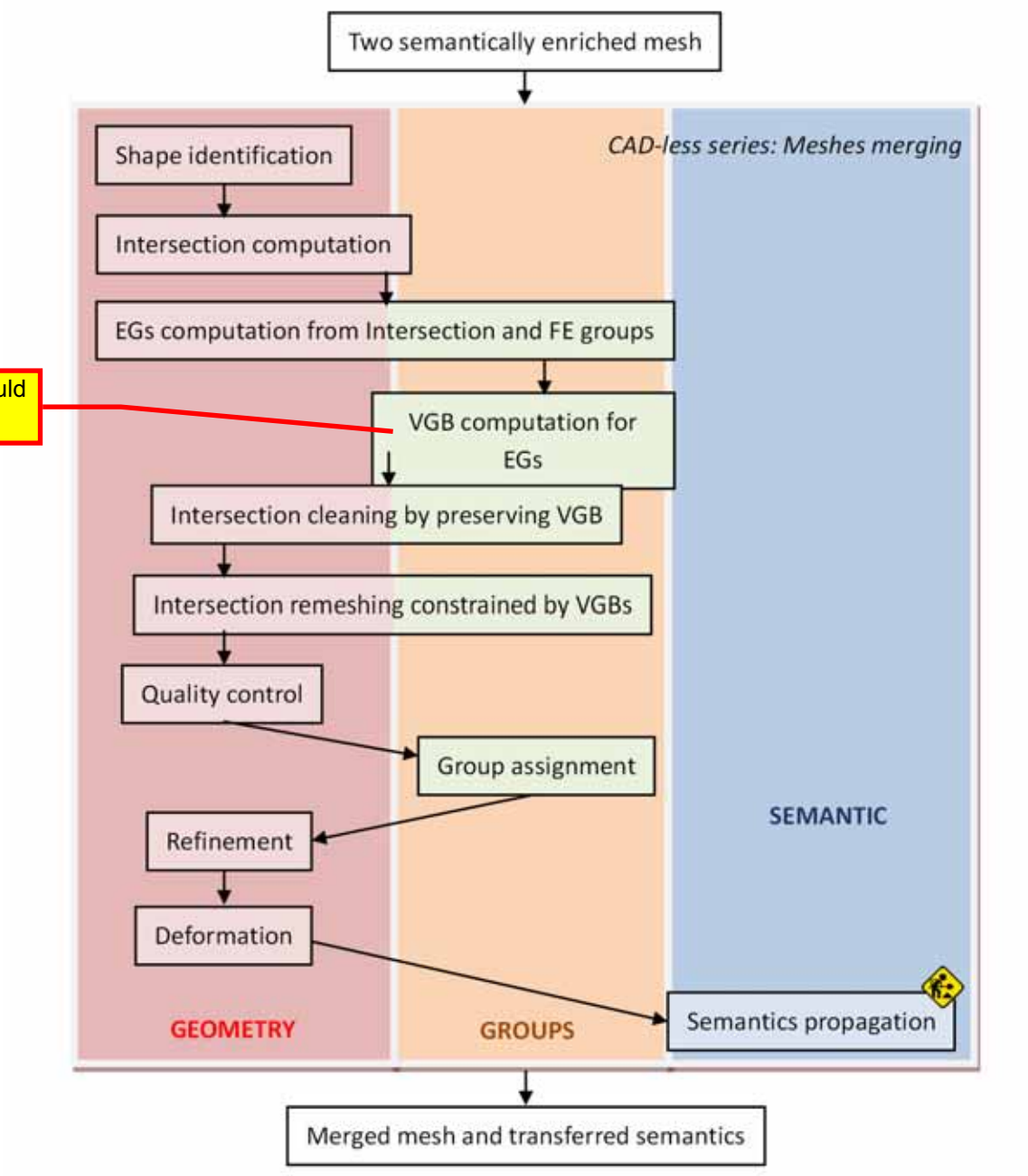
The workflow and necessary tools working at the geometric, groups and semantics levels is shown in the figure 6.4. The tools are used successively and categorised into three categories corresponding to the three columns.

6.2.2 Intersection computation

The computation of intersection between two meshes is decomposed into

- Identification of triangles potentially in contact,
- Computation of intersection nodes and creation of intersection curves,
- Intersection curves optimisation.

These steps are detailed separately in following paragraphs.



The sentence should be in one line.

Figure 6.4: Workflow and necessary tools for merging meshes

6.2.2.1 Contact faces detection

This step aims at finding the faces that are potentially in contact. It consists in the detection of intersections between scaled bounding boxes built on each face of $Mesh1$ and $Mesh2$ [23]. The bounding box of a face is computed along the parametric directions and scaled with an empiric factor of 1.05 to avoid inaccuracy problems. It results that the number of detected faces in contact can be greater than the number of real couples of intersecting faces. The outputs of this step are two sets F_1 and F_2 containing respectively some faces of $Mesh1$ and $Mesh2$. The i^{th} face of $F1$ is potentially in contact with the ith face of $F2$. Obviously, since a bounding box of $Mesh1$ may intersect several bounding boxes of $Mesh2$, and vice-versa, $F1$ and $F2$ can contain several times the same face. The following figure 6.5 shows an example of contact faces detection applied on two triangles A and B. A bounding box is computed for each triangle, the intersection between the two bounding boxes is verified and the potential contact between the two triangles A and B is confirmed.

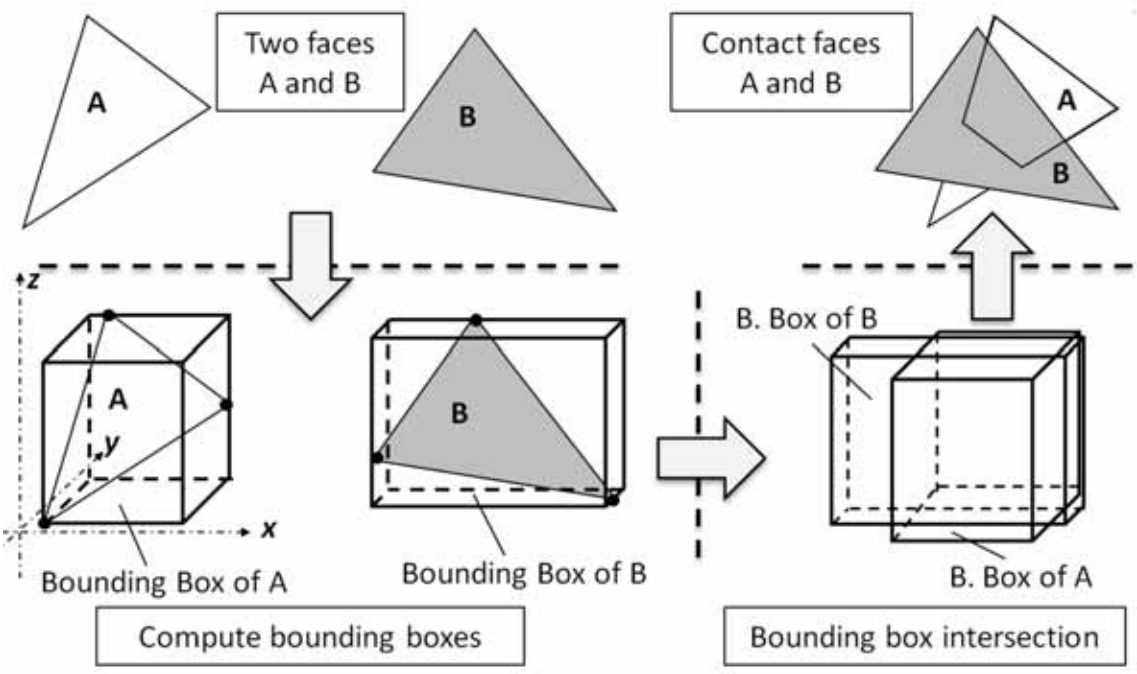


Figure 6.5: Contact faces detection using Bounding Box intersection

6.2.2.2 Intersection curves definition

For each couple of faces ($f1$, $f2$), the intersection nodes between the edges of $f1$ and the face $f2$ as well as the intersection nodes between the edges of $f2$ and the face $f1$ are computed. This algorithm is run over the entire sets F_1 and F_2. It results in two sets N_{12} and N_{21} of so-called edge intersection nodes (fig.6.6.a_1 and fig.6.6.a_2). See how F_1 and F_2 have been built, some couples of faces contained in F_1 and F_2 may finally not intersect. Thus, the two arrays are updated so that they solely contain the faces really intersecting.

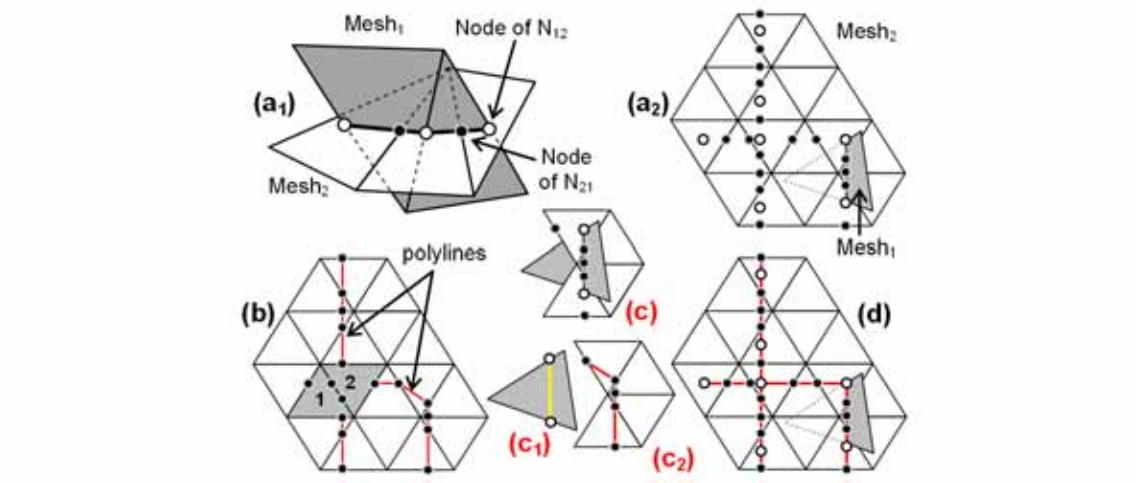


Figure 6.6: Intersection curve construction

The intersection curve construction consists in the definition of the edges connecting the initially isolated intersection nodes. It follows two steps:

1. **Local step**: creation of edges between edge intersection nodes located on faces on which there are exactly two edge intersection nodes (fig.6.7.b). Since the faces 1 and 2 own more than two edge intersection nodes, the creation of the connections is postponed to the second step. This is performed independently on the two meshes as illustrated on the two figures 6.7.c_1 and 6.7.c_2 detailing the local configuration of figure 6.7.c. Thus, it gives rise to two sets of potentially disconnected poly-curves (a set for each mesh);

2. **Global step**: insertion of the poly-curves of $Mesh1$ into the poly-curves of $Mesh2$. Depending on a set of identified configurations, the nodes of the

poly-curves of $Mesh1$ can be inserted in the poly-curves of $Mesh2$. It may require the creation of new edges. All the nodes are inserted but not all the edges. Actually, solely those edges whose end points are located in the same triangle are preserved. Thus, we obtain a set of poly-curves defined by multiple branches (fig.6.7.d).

Actually, this two-steps procedure is mandatory when the densities of the two meshes are too different. In this case, it is not possible to build directly all the connections after having merged the whole set of edge intersection nodes.

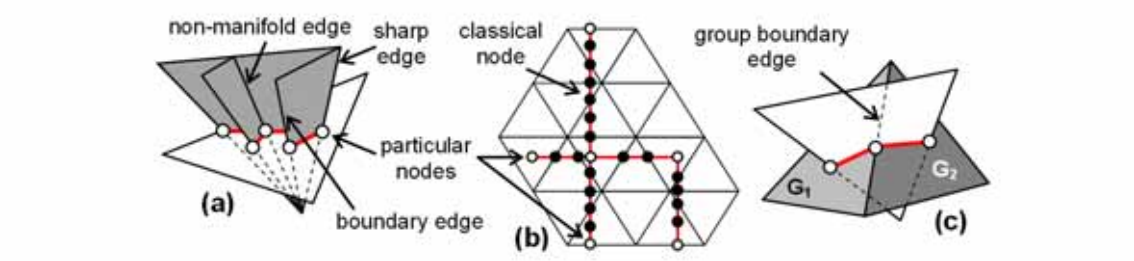


Figure 6.7: Intersection curve construction

6.2.2.3 Intersection curve optimisation

As shown on the result of figure 6.3.b, the intersection between two meshes may lead to an intersection poly-curve defined by a non-uniformly distributed set of nodes. This situation is even more amplified when the mesh densities are significantly different around the contact interface. Since this poly-curve is used as a basis for the re-meshing of the contact interface, it is mandatory to improve the distribution of nodes such that the inserted triangles are as little degenerated as possible. The optimisation works in two steps:

1. so-called **particular nodes** of the intersection curves are tagged so that they cannot be deleted or even moved during the optimisation; These particular nodes belong to either non-manifold edges, or mesh boundary edges, or group boundary edges or sharp edges (fig.6.7.a and fig.6.7.c). Figure 6.7.b shows the result of this step on the previous example. Figure 6.7.c shows an example where two groups of faces $G1$ and $G2$ have been defined. Thus,

the node arising from the group boundary edge has to be kept to maintain the semantics potentially associated to the groups;

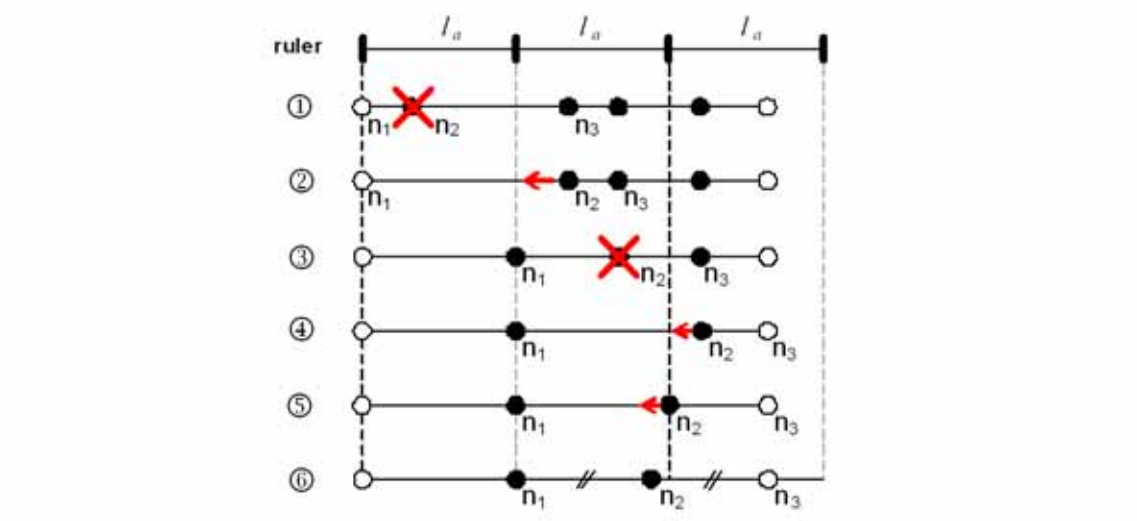


Figure 6.8: Intersection branch optimisation according to the ℓ_a average length criterion, the nodes n_1 and n_6 are considered as particular nodes

2. **deletion of "classical" nodes**, i.e. those nodes that have not been tagged in the previous step, according to the edge average length criterion initialised while computing the average length ℓ_a of the edges of the triangles that have been identified in the array F_1 and F_2. The overall algorithm works successively on all the branches of the intersection curve. The extremities of these branches can either be an extremity of an intersection curve, or an edge intersection node on non-manifold edges, or an edge intersection node located on edges to be kept. Figure 6.8 illustrates the treatment of one branch. It shows a ruler whose basic unit is equal to ℓ_a and the steps of the optimisation process. To ease the understanding of the algorithm, the poly-curve is here deployed and the nodes are spread according to the curvilinear position they have onto the 3D poly-curve. The ruler is located so that its left extremity matches the 1^{st} extremity. The algorithm optimises iteratively a segment of two connected edges that are composed by three nodes. If the distance between the first node and the middle node is greater than the average length ℓ_a, the middle node is moved toward the

first node; otherwise the middle node is removed. If the third node corresponds to another extremity of the optimised branch, the middle node is just moved to the midpoint of the 1^{st} and $3rd$ nodes. The pseudo-code can be detailed in algorithm 12.

Algorithm 12 Optimisation of the intersection curves

Begin
$n_1 \leftarrow 1^{st}$ extremity
$n_2 \leftarrow$ neighbour node of n_1
while $n_2 \neq 2^{nd}$ extremity **do**
  $n_3 \leftarrow$ neighbour of n_2 but not n_1
  if distance(n_1, n_2) $< \ell_a$ **then**
    n_2 is deleted
    $n_2 \leftarrow n_3$
  else
    n_2 is moved to along $[n_1, n_2]$ so that distance(n_1,n_2) $= \ell_a$
    $n_1 \leftarrow n_2$, $n_2 \leftarrow n_3$
  end if
end while
$n_3 \leftarrow n_2$, $n_2 \leftarrow n_1$, $n_1 \leftarrow$ neighbour of n_2 not n_3
if distance(n_1, n_2) + distance(n_2, n_3) $< 1.5 \times \ell_a$ **then**
  n_2 is deleted
else
  n_2 is moved to middle of $[n_1, n_3]$
end if
End

To optimise this process, and thus obtained equidistant nodes, one could imagine a second iteration that would move the remaining nodes according to a new target length obtained while computing the average length of the edges remaining in the poly-curve.

6.2.3 Intersection zone re-meshing

Once the intersection has been identified and the intersection curves have been computed, the next stage consists in re-meshing this intersection zone by taken into account the intersection curves and the enriched semantics. To achieve this objective, three steps are listed and detailed in the following subsections:

- Re-meshing zone definition in a bandwidth around the intersection curves,
- Elementary patches identification and elementary holes generation,
- Filling different elementary holes and updating the semantics.

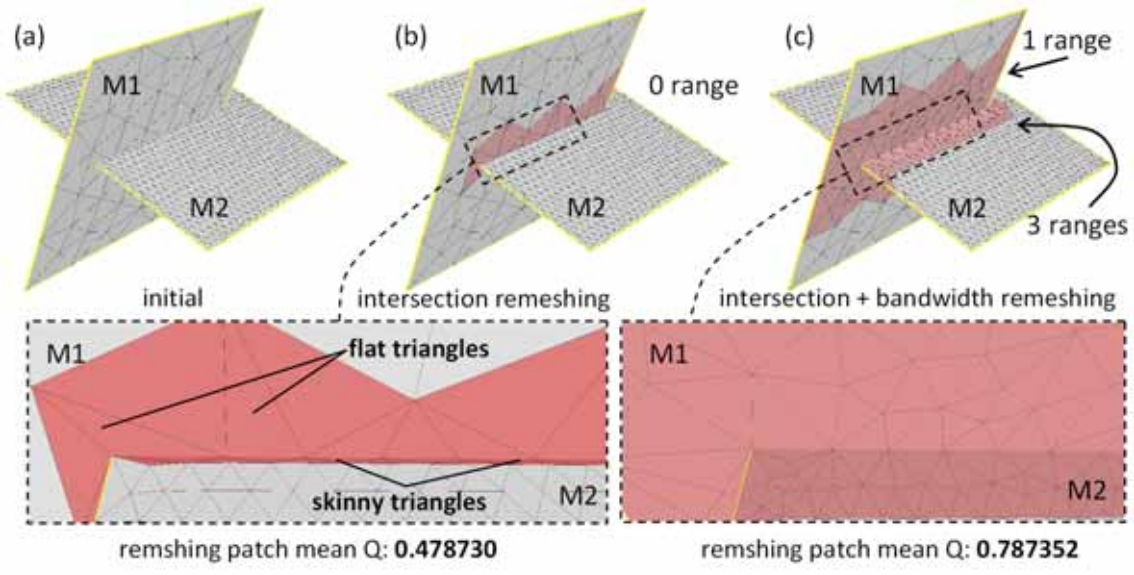


Figure 6.9: Two intersection meshes with different density (a), Re-meshing of intersection faces (b) and re-meshing of intersection faces plus their neighbourhood of different bandwidths (c)

6.2.3.1 Re-meshing zone definition

To prepare the generation of triangles as equilateral as possible, not only the triangles involved in an intersection interface have to be deleted, but also those which are in its more or less close neighbourhood. This is especially true when the two meshes that have to be merged own very different densities. Figure 6.9 shows some experimentation results where several values for the bandwidth are tested. Figure 6.9.a shows two intersecting meshes that have very different densities, approximately triangles in the mesh $M1$ are 5 times bigger than the ones in the mesh $M2$. The re-meshing process acting only on the intersecting triangles is shown on Figure 6.9.b. The re-meshing process acting on the intersecting triangles

and their surrounding neighbours is shown in figure 6.9.c. A zoom of each picture is also proposed below. Without the use of the intersecting triangles neighbours, there are many skinny triangles produced on $M2$ since the intersection curve is very close to the re-meshing zone boundary. The $M1$ has also very flat triangles since the density of the intersection nodes is much higher than the mesh nodes density. Therefore, the transition between original mesh $M1$ and the re-meshed part is not good. Comparing the mean quality of the re-meshed zone, the re-meshing of intersection in a bandwidth (fig.6.9.c) produces elements of quality ($Q = 0.787$) much better than the ones ($Q = 0.478$) produced by re-meshing of solely the intersecting triangles (fig.6.9.b).

For a given intersection curve, potentially defined by one or several branches, the notion of neighbourhood is therefore defined increasingly as follows:

- the **first neighbourhood** of the poly-curve, or the neighbourhood at the first order, gathers together all the triangles of the contact interface, i.e. all the triangles contained in F_1 and F_2;
- the **second neighbourhood** of a poly-curve gathers together all the triangles defined by at least one node connected to at least one triangle of the first neighbourhood;
- the $i^{th} neighbourhood$ of a poly-curve gathers together all the triangles defined by at least one node connected to at least one triangle of the $(i-1)^{th}$ neighbourhood of the poly-curve. If a poly-curve has no i^{th} neighbourhood, there will be no neighbourhood of higher order.

Given these definitions, it is easy to identify the various neighbourhoods of the intersection curves in order to collect triangles defining the re-meshing zone. Thus, to collect the faces included in a n-ring neighbourhood of N_b triangles around an intersection curve ($N_b \geq 1$), all the triangles of the neighbourhoods having an order varying from 1 to N_b are recursively collected. The result of figure 6.3.d_1 has been obtained while deleting the triangles in the first neighbourhood of the intersection curve ($N_b = 1$) for the half-vase and in the second ($N_b = 2$) for the half-cylinder. The result of figure 6.9.c has been obtained by re-meshing

the triangles zone in the first neighbourhood of the intersection curve ($N_b = 1$) for the $M1$ and in the third ($N_b = 3$) for the $M2$.

To automatise the instantiation of the control parameter N_b, a specific algorithm is proposed. It takes into account both the ratio between the densities of the two meshes, which can be more or less equal, and the minimum distance from the intersection curve to the boundary of its first order neighbourhood, which can be more or less equal to the half average length $\ell_a/2$. Depending on whether the average lengths of the edges of the two meshes are almost equal or not, two different cases are distinguished. In the first case, i.e. average lengths almost equal, the triangles of both meshes are collected in a n-ring neighbourhood computed from the analysis of the smallest distance ℓ_{min} from the intersection curve to the boundary of its first neighbourhood. In the second case, the triangles of the mesh of greatest density are removed in a n-ring neighbourhood equal to 1, whereas the triangles of the mesh of smallest density are collected in a n-ring neighbourhood defined from the ratio between the half average length $\ell_a/2$ and the smallest distance ℓ_{min}. This algorithm does not prevent the creation of asymmetric configurations but it tries to minimise them.

The algorithm is written in pseudo-code (algo.13). The "Round[x]" operator returns the closest integer to the real x. The function "Collect[$mesh$, F, n]" collects all the triangles of $mesh$ included in the neighbourhood of order n of the intersection curve whose first neighbourhood is defined by F. When computing ℓ_{min}, the nodes of the intersection curve that are on the boundary of the contact interface are not considered.

6.2.3.2 Re-meshing zone cleaning

After having defined the re-meshing zone in a bandwidth around the intersection curves the re-meshing should be performed under several constraints:

- The re-meshing zone contour
- The intersection curves
- The virtual group boundaries (VGB defined in section 5.2.2 on page 154)

Algorithm 13 Collection of the n-ring neighbourhood for re-meshing zone

Begin

Compute the two average edge lengths, respectively ℓ_{1a} and ℓ_{2a}, of the triangles respectively contained in F_1 and F_2 (section 6.2.2)
$N_b \leftarrow$ Round $\left[\max[\ell_{1a}, \ell_{2a}]/\min[\ell_{1a}, \ell_{2a}]\right]$
if $N_b = 1$ **then**
  Compute ℓ_{min} the smallest distance from the intersection line to the boundary of its first neighbourhood
  $N_b \leftarrow$ Round $\left[\max[\ell_{min}, \ell_a/2]/\min[\ell_{min}, \ell_a/2]\right]$
  $N_b \leftarrow \min[N_b, 2]$
  Collect [Mesh1, F1, N_b]
  Collect [Mesh2, F2, N_b]
else
  if $\max[\ell_{1a}, \ell_{2a}] = \ell_{1a}$ **then**
    Collect [Mesh1, F1, 1]
    Collect [Mesh1, F1, N_b]
  else
    Collect [Mesh1, F1, N_b]
    Collect [Mesh1, F1, 1]
  end if
end if

End

For applying this constrained re-meshing process correctly and easily, we decompose at first the re-meshing zone into several elementary patches according to the different VGBs. These different patches are computed from the decomposition of elementary groups (notion of EG introduced in section 5.2.4 on page 160) by using the re-meshing zone and different groups (middle-level semantics). For the intersection zone re-meshing, we are only interested in the elementary patches having relation with the re-meshing zone. Obviously each elementary patch could be related with the groups to which its mesh elements are belonging to.

Once different elementary patches are identified, the collected triangles in each patch could be removed by keeping the patch contour edges. At this stage, different elementary holes are generated. Before filling new mesh elements we still have to take into account the intersection curves that lie on these elementary holes by splitting the elementary holes. Since the re-meshing zone is defined in

the neighbourhood of the intersection curves so that the re-meshing contour is not really connected with the intersection curve extremities. Starting from the identified and optimised intersection poly-curves, the extremity nodes are first projected onto the elementary hole contour (closest point algorithm) so that all the intersection curves are going through the elementary holes. Therefore the re-meshing zone is decomposed into several elementary holes to mesh separately without any other constraints. Here, it is important to recall that without the decomposition into EGs, it is impossible to know what are the holes to be filled in.

On the example of figure 6.10.a, the re-meshing zone is decomposed into two elementary patches A and B due to two triangle groups and the triangles inside are removed. The two extremity nodes n_1 and n_2 are projected and new edges e_1 and e_2 are inserted to extend the intersection curve in touch with the re-meshing zone contour. Then, the algorithm goes through the poly-curves data structure to create subsets of poly-curves forming the contour of subdivided holes. As a consequence the two holes A and B are subdivided in four sub-holes $A1$, $A2$, $B1$ and $B2$ (fig.6.10.b).

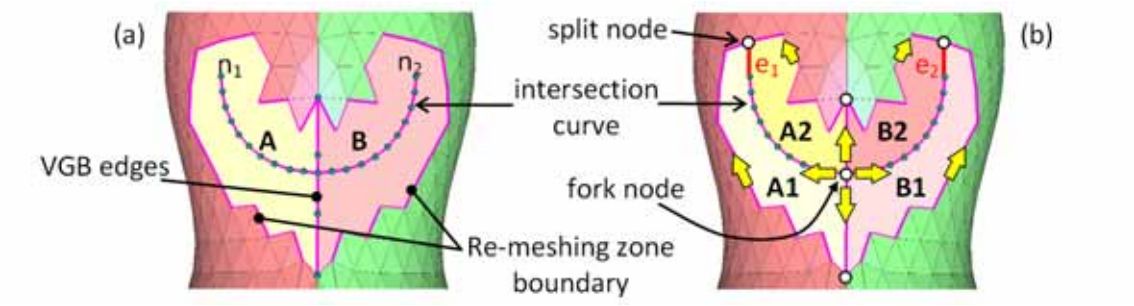


Figure 6.10: Subdividing two elementary holes A and B (a) into four elementary holes (b)

6.2.3.3 Filling holes and updating the semantics

For filling different elementary holes we use the triangulation, mesh refinement and relaxation that are presented in chapter 4 (p.85). We use the adapted and modified version of the Liepa's algorithm [60] to answer the FE requirements (see section 4.4.1 (p.121) for more details on the algorithm and improvements

that have been done). Given a hole contour, if the algorithm builds iteratively a triangulation so that its area is minimum at each step the resulting triangles are degenerated and some others are flat (fig.6.11.a). Whereas if this algorithm builds iteratively a triangulation so that the aspect ratio Q is maximised the resulting triangles are better (fig.6.11.b).

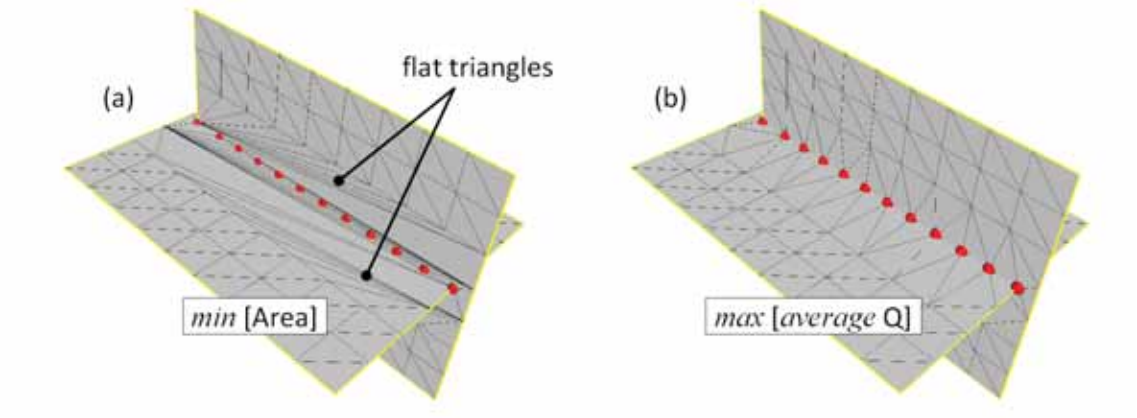


Figure 6.11: Comparison between the minimum area (a) and the maximum aspect ratio (b) triangulation criteria

In figure 6.3.e, the hole-filling algorithm maximising the mean quality is applied on the 6 elementary holes. At this stage the associated semantics is immediately recovered. Therefore for each filled elementary hole, the groups that were defined on the deleted mesh elements are immediately defined on the newly created triangles (fig.6.3.f). Here, the semantics available on the initial mesh are maintained in the resulting mesh. Obviously, this operator does not cover all the needs relative to the manipulation of semantics when merging two FE meshes.

To ensure both a good quality of the triangles with respect to the FE criteria, and a smooth blending of the modified area with the surrounding mesh, two additional steps are required:

- **insertion of new nodes** at the centroid of some triangles and swap of edges for each elementary patch so that the optimised triangulation approximates the density of the surrounding meshes. This is especially interesting when the two meshes own triangles of heterogeneous sizes in the filled area. The details of the nodes insertion algorithm are presented in [60].

- **deformation** of the re-meshed zone by fixing all elementary patch boundaries so that the connections between the inner and surrounding meshes satisfies either position, tangency or curvature blending conditions. It also relaxes the position of the nodes so that better triangle shapes are obtained. The adopted deformation tool for relaxing meshes has been presented in section 4.5 (p.129). This deformation tool allows imposing shape constraints on the nodes. Actually, the shape recognition (section 4.1 on page 86) is launched on the models before the merging process so that basic shapes (e.g. plane, sphere and cylinder) can be saved inside new groups. Therefore, during the relaxation step of the merging process, the inserted nodes can be constrained to stay on a special shape to keep the initial model shape.

During the two mesh optimisation process the re-meshing zone boundary, intersection curves and group boundaries are not changed. Figure 6.3.f shows the nodes insertion onto the newly triangulated patches and these patches are relaxed by using the deformation engine (fig.6.3.g).

Thus, following this section 6.2.3, the holes are filled in while satisfying both a criterion relative to the shape of the deformed triangles (sizes and orientations) and a criterion relative to the quality of the blending between the inserted and surrounding meshes. More precisely, our approach produces a deformation that minimises the curvature variation over the mesh and satisfies potentially identified shape constraints.

6.2.4 Experimentation results

This triangle mesh merging operator has been applied on several academy examples and three of them are shown in figure 6.12. The first example shows the merging of the scanned model of a hand with the scanned model of a thumb (fig.6.12.a). This result shows that using appropriate thresholds to decide whether the nodes of the intersection line have to be kept or not (section 6.2.2.3), the risk to obtain aliasing effects along the intersection line are minimised. The second example illustrates a configuration where the two meshes intersect smoothly, i.e. with no sharp intersection (fig.6.12.b). The third example is used to illustrate how

the developed approach can be used to merge non-manifold meshes (fig.6.12.c). First, a part of a sphere is merged with a plane thus producing non-manifold edges along the intersection line. This model is then merged with a second plane. Table 6.1 (on page 201) gathers together the analysis of the results with respect to the aspect ratio before and after the merging operation.

The introduced triangle mesh merging operator has also been tested and validated on an EDF's engineering project. More precisely, the algorithm is applied on a caisson which had to be rigidified by inserting several stiffeners in the appropriate directions. Figure 6.13.a represents a possible solution to solve this problem. Both the static and dynamic behaviours of the structure are improved. Figure 6.13 shows how the multiple input meshes (fig.6.13.a) are treated to obtain a single mesh while keeping intact the semantic information (fig.6.13.e). The different colors represent the group assignments. The algorithm first detects the contact interfaces, computes the intersection lines, optimises the number of intersection nodes and cleans the contact interface (fig.6.13.b and fig.6.13.c). Here, the meshes are of homogeneous sizes and the cleaned area has a n-ring neighbourhood of one. Thus, for each newly inserted stiffener, 22 holes appear on the caisson and 7 on the stiffener. These holes are then filled using the newly developed quality criterion (fig.6.13.d) and the group assignments are maintained onto the newly created elements (fig.6.13.e). The nodes are finally relaxed using our deformation engine.

To better analyse the quality of the resulting mesh in comparison with the initial mesh, table 6.1 is shown below. The computation of aspect ratio for triangles is presented in the section 4.3 (p.120). The aspect ratio is between 0 and 1. Mean aspect ratio for all triangles before and after merging is given in the table. The minimum and maximum of aspect ratio among all triangles are also listed in the table. One can notice that this operator do not affect the quality of the meshes in terms of their aspect ratios, sometime the merged meshes even have better aspect ratios.

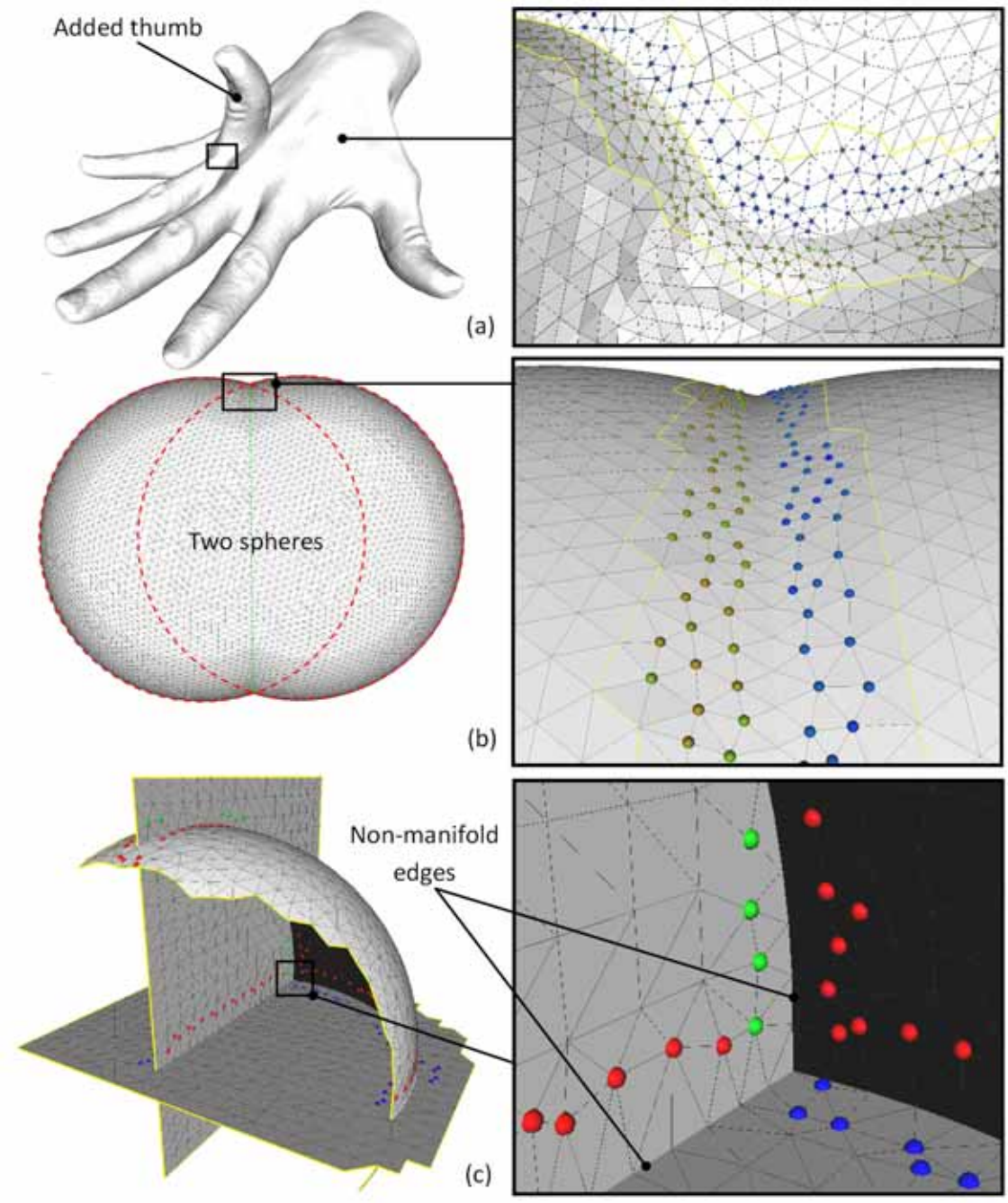


Figure 6.12: Merging of two scanned models (a, courtesy of MPII), two spheres intersecting smoothly (b) and an example of non-manifold configuration (c).

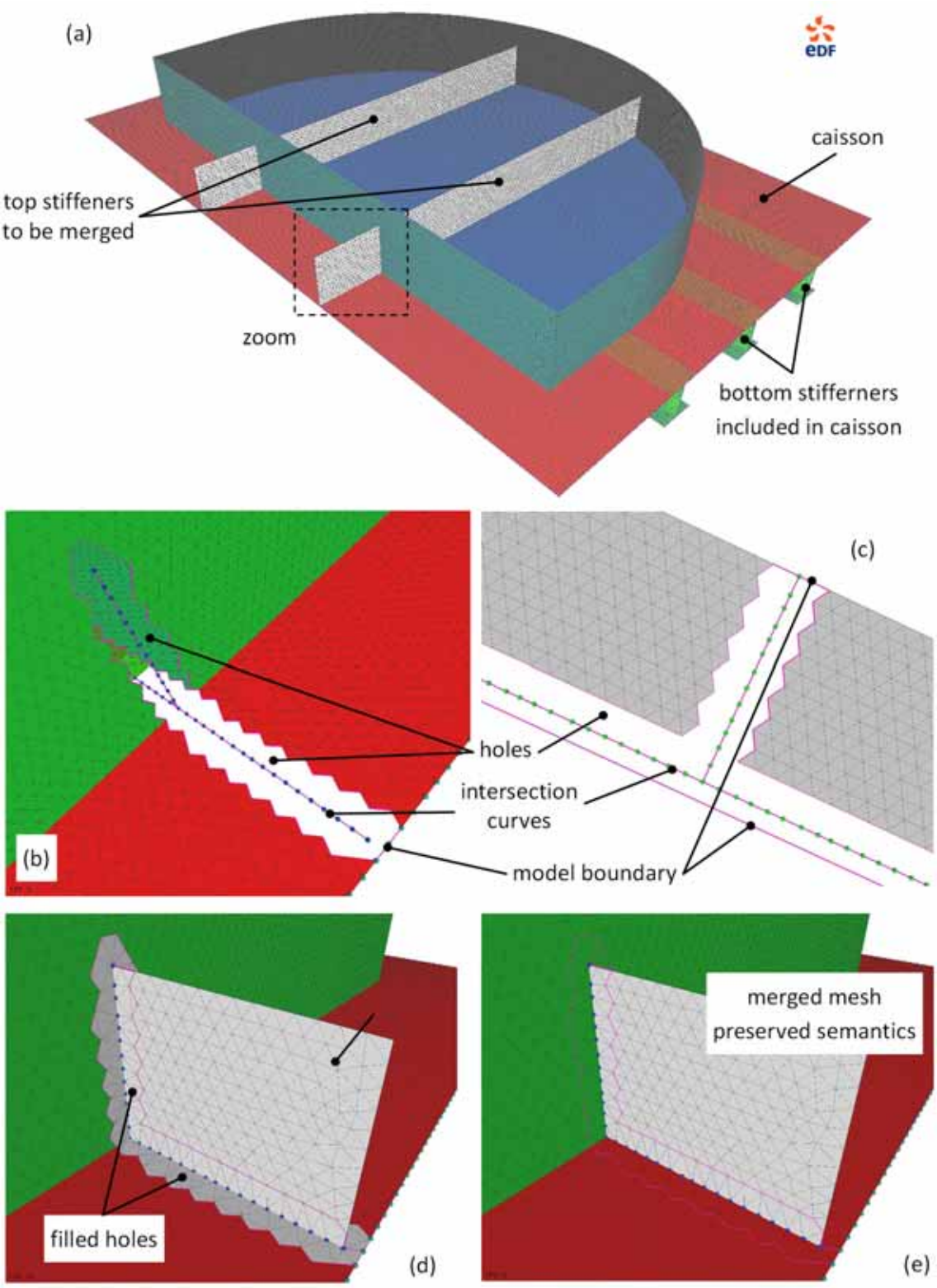


Figure 6.13: Overall merging approach on the example of two stiffeners that have to be merged with a caisson model. (courtesy EDF R&D)

Aspect ratio Q		Vase fig.6.3	Hands fig.6.12.a	Spheres fig.6.12.b	Spheres fig.6.12.c	Caisson fig.6.13	Stones fig.6.14
Initial	Min	0.319	0.001	0.882	0.458	0.003	0.005
	Max	0.999	0.999	1.000	0.992	0.999	0.995
	Mean	0,880	0,858	0.927	0.751	0.972	0.638
Merged	Min	0.204	0.007	0.317	0.278	0.003	0.005
	Max	0.999	0.997	1.000	0.998	0.999	0.999
	Mean	0,859	0,800	0.910	0.760	0.968	0.664

Table 6.1: Comparison of the aspect ratio Q for various merged models

6.2.5 Mesh merging in face/face mode

The merging process in face/edge mode presented just before could be easily extended to the face/face mesh merging mode. Actually, most of the elementary algorithms developed for the face/edge merging mode can be adapted to treat the face/face mode (fig.6.14). In this case, no intersection curves are computed. However, the transfer of semantic information potentially attached to these two meshes is much more difficult in the face/face mode than in the face/edge one. This aspect has not been treated yet.

The main steps are:

1. **Contact faces identification** by using the intersection of bounding boxes,
2. **Contact faces deletion** in a bandwidth for having enough space to produce good quality mesh elements,
3. **Creation of a blending mesh** part for connecting the hole contour of the first mesh and the hole contour of the second mesh.

Figure 6.14 shows an example of face/face merging process performed on two meshes that are acquired from two real stones via laser scanner. The contact triangles are computed at first stage (fig.6.14.a) between the two meshes. An additional view of two meshes separated is also shown in figure 6.14.b. Then the contact zone is defined and deleted in a bandwidth with intersection triangles and their 2^{nd} neighbourhood. Finally, the mesh patch is created and connects the two hole contours (fig.6.14.d).

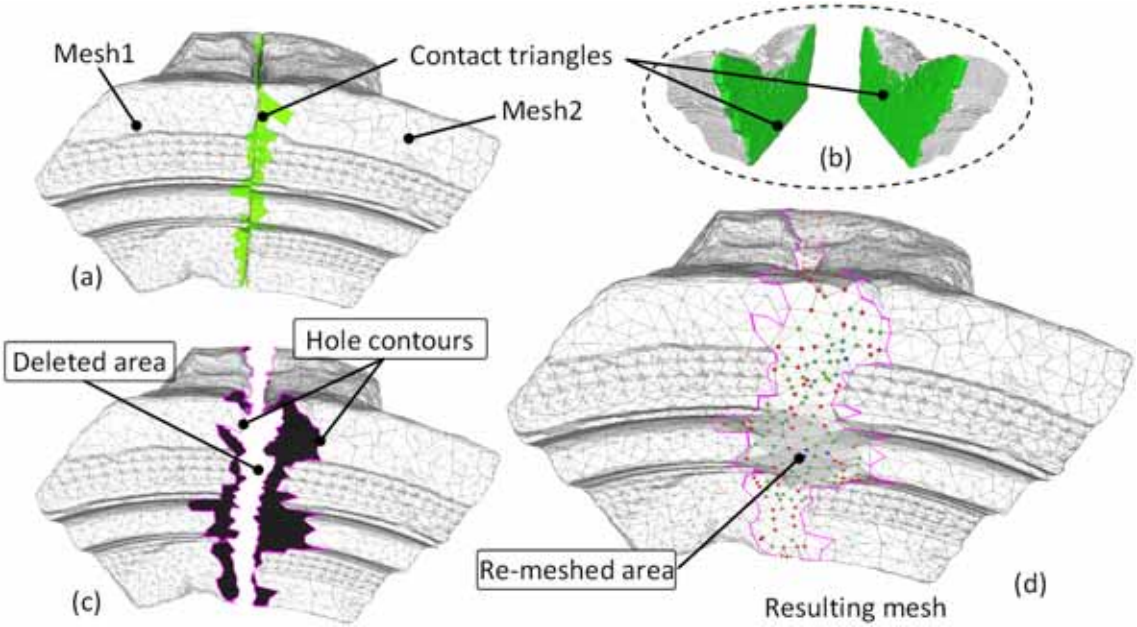


Figure 6.14: Two scanned stones merged using an approach similar to the face/edge mode

6.3 Mesh cracking operator

In this section, the mesh cracking operator proposed in [72, 74] is detailed. This operator aims at directly modifying a semantically enriched FE mesh. In our work, we do not simulate the physical phenomenon of crack propagation and so on [51] since we stay at the level of the geometric models enriched with semantic information. Here, a "geometric" crack is inserted inside the volume of a structure. It results in a dividing the mesh into two sub-meshes having two coincident internal faces modellilng the crack. Actually, cracks are usually represented as closed and so having no volume. The surfaces representing the two sides are distinct but coincident so that the nodes on the opposite sides of the crack have identical coordinates. Actually, this is performed while duplicating overlapping mesh entities (double nodes and face elements). Here the planar crack operator is presented. This crack is supposed closed at the initial instant $t_0 = 0$.

The principle of the crack operator is illustrated on figure 6.15. Initially a semantically enriched mesh model is shown in figure 6.15.a. There are two groups $G1$ and $G2$ and the crack feature to be inserted should follow the crack surface

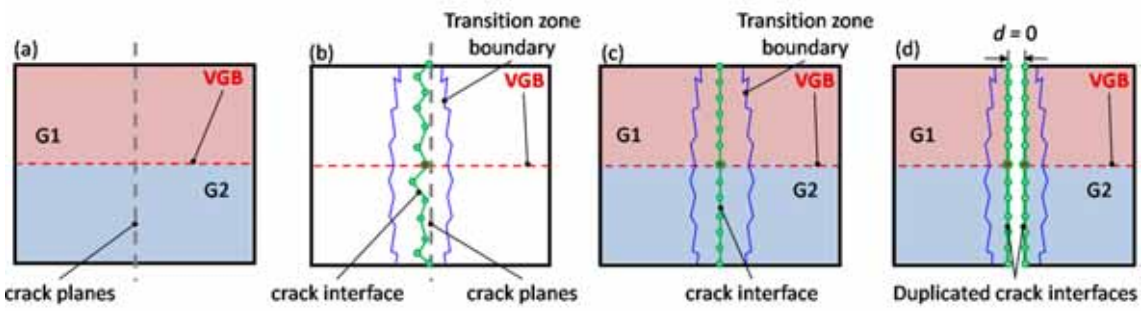


Figure 6.15: CAD-less mesh crack operation schema

(here it is a plane). The resulting mesh with the crack feature is shown in figure 6.15.b in which the model is roughly subdivided into two parts along the crack plane. Therefore a rough crack interface along the crack plane can be identified. The surface part following the crack surface is called crack interface and this surface on the two subdivided parts should have exactly the same tessellation. Therefore the nodes on the crack interface surface of one sub-part should have a corresponding node with same position on the other sub-part crack interface. Then this rough crack interface is deformed to match the crack plane (fig.6.15.c) and it is duplicated into two instances at the end (fig.6.15.d). During the deformation process, to avoid bad quality mesh elements, not only the interface surface but the other mesh elements are deformed. In addition, the modification should be local in order to keep the validity of the tuned mesh. Therefore, a transition zone is defined to limit the deformation zone. Within the transition zone the deformation is constrained to preserve several shape characteristics like model boundaries, group boundaries, etc.

As mentioned in subsection 4.4.3 (p.126), the duplicated mesh elements correspond geometrically to the introduction of what we call the "contact zone" in the model. Spatially, the model is continuous (no gaps) but topologically the model is split into two sub parts. Therefore, the two sub-parts are considered "in contact" and the duplicated part is the "contact zone". Thus, the crack operator discussed in this section can also be used to introduce a contact interface in the case of planar surfaces in contact. The contact zone (crack feature) to be inserted could be limited by a profile (e.g. circle, rectangle etc.) so that the crack could do not go through the whole model.

6.3.1 Overall process of planar crack insertion into semantically enriched meshes

Figure 6.16 shows the workflow of the crack operator and tools necessary in terms of geometry, groups and semantics. The input is a semantically enriched mesh. The first stage consists in identifying the shape primitives and VGBs for the exiting groups. Then, a sequence of geometric operations follows: rough crack interface computation, crack interface pretreatment, transition zone definition, constrained deformation limited in the transition zone and crack interface duplication. At the end, the semantics propagation is foreseen in the future.

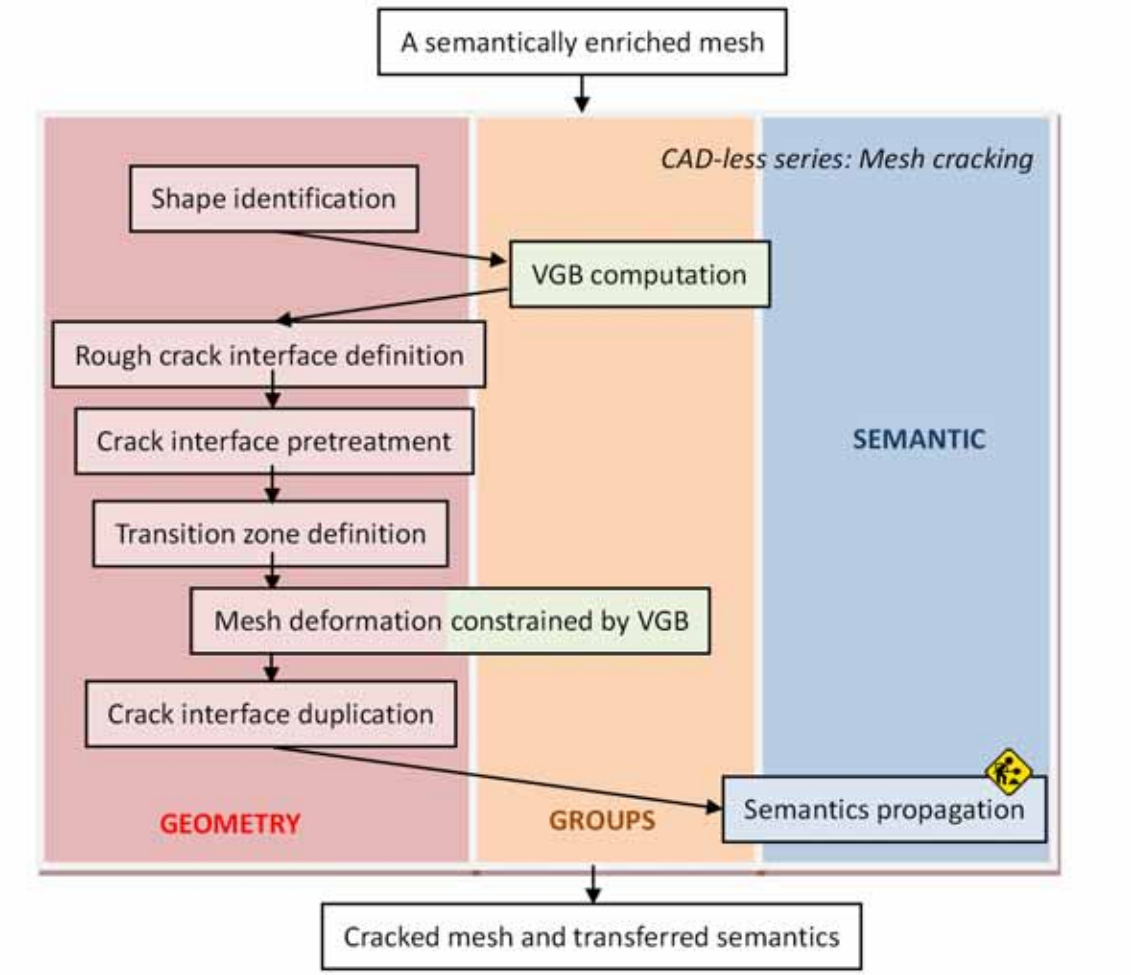


Figure 6.16: Workflow and necessary tools for the crack operator

The **crack interface** is a set of triangles (resp. edges) in case of 3D mesh (resp. 2D mesh). It corresponds to the boundary of the tetrahedra (resp. triangles) lying completely in one side with respect to the crack plane and that has to be deformed to respect this plane. The nodes relative to the interface elements are forced to move onto the crack plane. To avoid degenerated triangles/tetrahedra, some elements not directly in contact with the crack plane are also moved. The deformation process is based on the one presented in section 4.5 (p.129). Finally, the crack interface is created by duplicating nodes and faces on the two sides of crack.

These steps are detailed in the following subsections. For sake of clarity, the planar crack operation is bounded by a circle on all the figures. An example is shown on figures 6.17 and 6.23. It's a tetrahedral mesh on which three groups of tetrahedra $G1$, $G2$ and $G3$ are defined (fig.6.17.a). The virtual group boundary of these groups is also computed with the tool presented in section 5.2 (p.152). Then, different basic shape information on the model boundary are identified (fig.6.17.b) by using the shape recognition tool presented in section 4.1 (p.86). Here it consists in a sphere surface and six planar surfaces. The model's volume is subdivided into two parts (fig.6.17.c) according to the crack plane and the two sub parts are shown separately (fig.6.17.d and fig.6.17.e). Figures 6.23.a and 6.23.b show the initial mesh on which different important nodes are identified. The model is deformed so that a set of nodes are moved onto the crack plane and their neighbour nodes are moved for relaxing the mesh by preserving the shape on the model boundary and group boundary (fig.6.23.c and fig.6.23.d). In the next sub-sections, the principle steps of the crack operator are detailed: mesh elements classification and crack interface identification (subsection 6.3.2), crack interface pretreatment (subsection 6.3.3), mesh deformation on the level of crack interface (subsection 6.3.4) and duplication of nodes and faces.

A data-structure for the crack operator is defined as following (algo.14): In the data-structure, the first two attributes (*point* and *normal*) are arrays of three real numbers used to define the plane equation. The mesh is an object of type Mesh that would be either triangle mesh or tetrahedral mesh. The two lists T_1 and T_2 store the triangles or tetrahedra of the two sub-parts. The list CI stores the interface elements.

Algorithm 14 Data strucutre of Crack

Structure Crack

$point(3)$	As **Real**
$normal(3)$	As **Real**
$mesh$	As **mesh**
T_1	As **List**
T_2	As **List**
CI	As **List**

End Structure

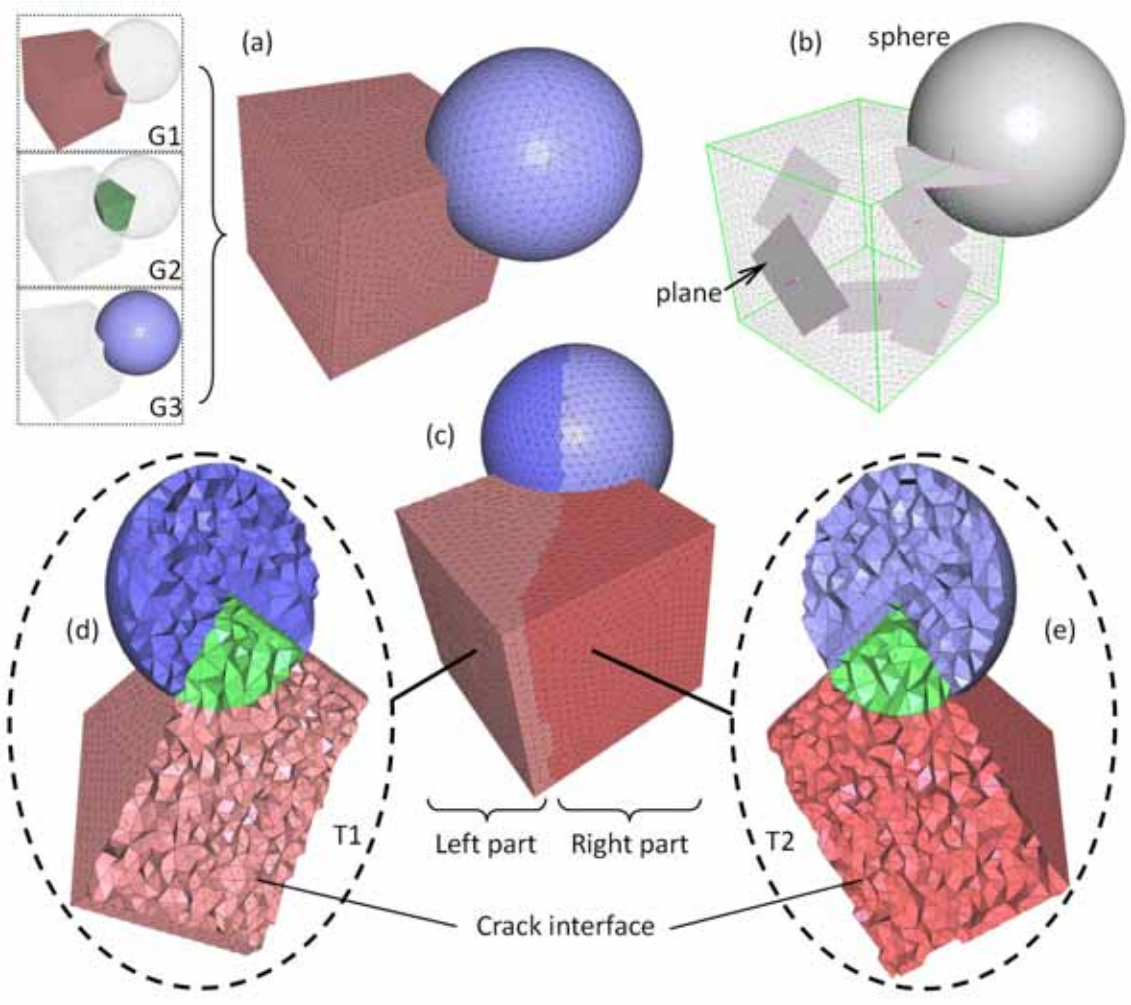


Figure 6.17: Rough crack interface computation over a 3D mesh containing three groups

6.3.2 Mesh element classification and Crack Interface identification

This first step separates all the mesh nodes into two sets (N_1, N_2) according to their positions with respect to the two half-spaces (P,N) defined by the crack plane.

In the case of 3D mesh, a set T_1 gathers together the tetrahedra having their 4 nodes belonging to the half-space P, respectively T_2 for tetrahedra having at least one node in the half-space N. Analogously, for 2D mesh, the set T_1 gathers together triangles whose 3 nodes belong to the half-space P, and respectively T_2 for triangles defined by at least one node belonging to the half-space N. Figures 6.17.c and 6.17.d show separately the identified sets T_1 and T_2 when applying a planar crack.

Now, the Crack Interface (CI) has to be identified. For 3D mesh, the CI is defined as a set of triangles f shared by one tetrahedron t_i in T_1 and one tetrahedron t_j in T_2. Analogously, in the case of 2D mesh, the CI is a set of edges e shared exactly by one triangle f_i in T_1 and one triangle f_j in T_2. Until this stage the crack interface elements are initially computed and the volume of the model is subdivided into two parts. Before using deformation for flattening the crack interface there are still some treatments to do on the crack interface for avoiding the potential degenerated tetrahedra/triangles (see the next section).

In the stage of subdividing a tetrahedral model into two sub-parts T_1 and T_2, the number of nodes belonging to the half-space P is fixed at 4 because the produced interface is "flatter" than the other parameters. An experimentation of using different numbers is shown in the figure 6.18. The tetrahedral mesh is subdivided roughly into two parts T_1 and T_2 by the same crack plane. The minimal number of nodes belonging to the half-space P for a tetrahedron aggregated to the sub-part T_1 is varying from 1 to 4. Each case is shown separately from figures 6.18.a to 6.18.d. The produced crack interface is the top surface of the sub-part T_1 and has less aliasing effects in case the number is defined as 4 (fig.6.18.d).

The algorithm 15 for this crack interface definition in a tetrahedral mesh is written below. The data-structure of $Crack$ used in the following algorithm is detailed previously (algo.14). The function "computeVectorByTwoPoints (p1,

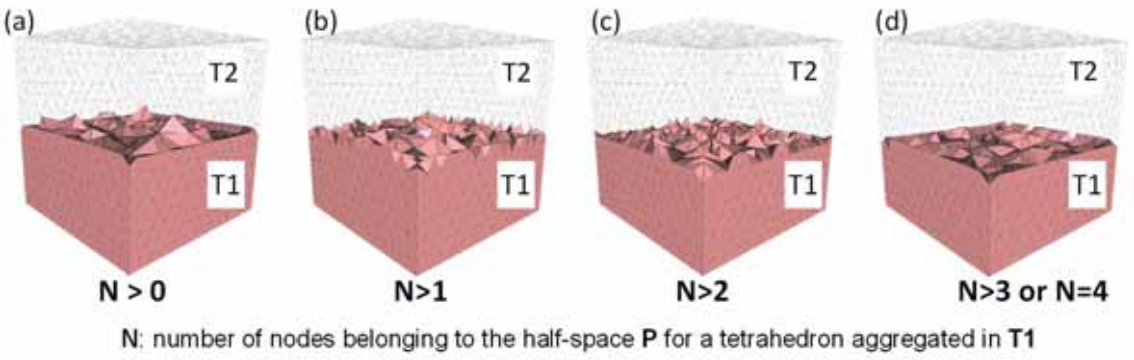


Figure 6.18: Rough crack interface identification

$p2$)" allows obtaining a vector from point $p1$ to point $p2$. The function "docProductByTwoVectors ($v1$, $v2$)" allows computing and returning the dot product between two vectors $v1$ and $v2$. The procedure "addElementToList (e,L)" allows adding element e into the list L. The function "isInList (e, L)" allows checking the existence of the element e in the list L, the return is true if L contains e.

Similarly, in case of a 2D mesh the number of nodes belonging to the half-space for a triangle aggregated to the sub-part T_1 is fixed at 3. The interface edges have less aliasing effects than the ones produced with smaller values. Therefore, a similar algorithm could also be written for the case of triangle mesh.

6.3.3 Crack Interface pre-treatment

After the initial crack interface identification, there may be some mesh elements (tetrahedra in 3D and triangles in 2D) associated with the crack interface that could become degenerate (null volume/area) after moving the crack interface nodes onto the crack plane. These problematic mesh elements are associating with usually more than 1 crack interface edges in 2D mesh or triangles in 3D mesh.

To easily understand the reason, the insertion of a crack in a 2D mesh is illustrated in the figure 6.19. The blue dashed line in figure 6.19.a represents the section of the crack plane so that the crack plane is perpendicular with the picture. The mesh is subdivided into two sub-parts T_1 and T_2 that are respectively gray and white. Therefore the red edges that are shared by two triangles

Algorithm 15 Crack interface definition over a tetrahedral mesh

```
Procedure crackInterfaceDefinitionTetrahedral(crack As Crack)
 1: Variable vector(3) As Real
 2: Variable n As Byte
 3: Variable list As List
 4: Variable t1,t2,tetrahedron As Tetrahedron
 5: Variable triangle As Triangle
 6: Variable node As Node
 7: for all node in crack.mesh do
 8:    vector ← computeVectorByTwoPoints(crack.point, node)
 9:    if dotProductByTwoVectors(crack.normal, vector) > 0 then
10:       addElementToList(node, list)
11:    end if
12: end for
13: for all tetrahedron in crack.mesh do
14:    n ← 0
15:    for all node In tetrahedron do
16:       if isInList(node, list) then
17:          n ← n + 1
18:       end if
19:    end for
20:    if n = 4 then
21:       addElementToList(tetrahedron, crack.T1)
22:    else
23:       addElementToList(tetrahedron, crack.T2)
24:    end if
25: end for
26: for all triangle in crack.mesh do
27:    if triangle is shared by 2 tetrahedra then
28:       t1 ← first tetrahedron
29:       t2 ← second tetrahedron
30:       if isInList(t1, crack.T1) xor isInList(t2, crack.T1) then
31:          addElementToList(triangle, crack.CI)
32:       end if
33:    end if
34: end for
End Procedure
```

of different sub-parts are considered as the interface. In this example the problematic triangles are the gray triangle tagged by a circle and the one tagged by a pentagram. The gray triangle tagged by a circle associates with three interface edges and the deformation process will place these three interface edges onto the crack plane. Figures 6.19.b, 6.19.c and 6.19.d show the evolution of this problematic triangle during the deformation. Effectively, at end this triangle will be completely flattened. The other problematic triangle tagged with pentagram associates with 2 interface edges that will be placed onto the crack plane by the deformation process. Figures 6.19.e, 6.19.f and 6.19.g show the evolution of this problematic triangle along the deformation progress. Similarly this problematic triangle will be flattened which is not acceptable.

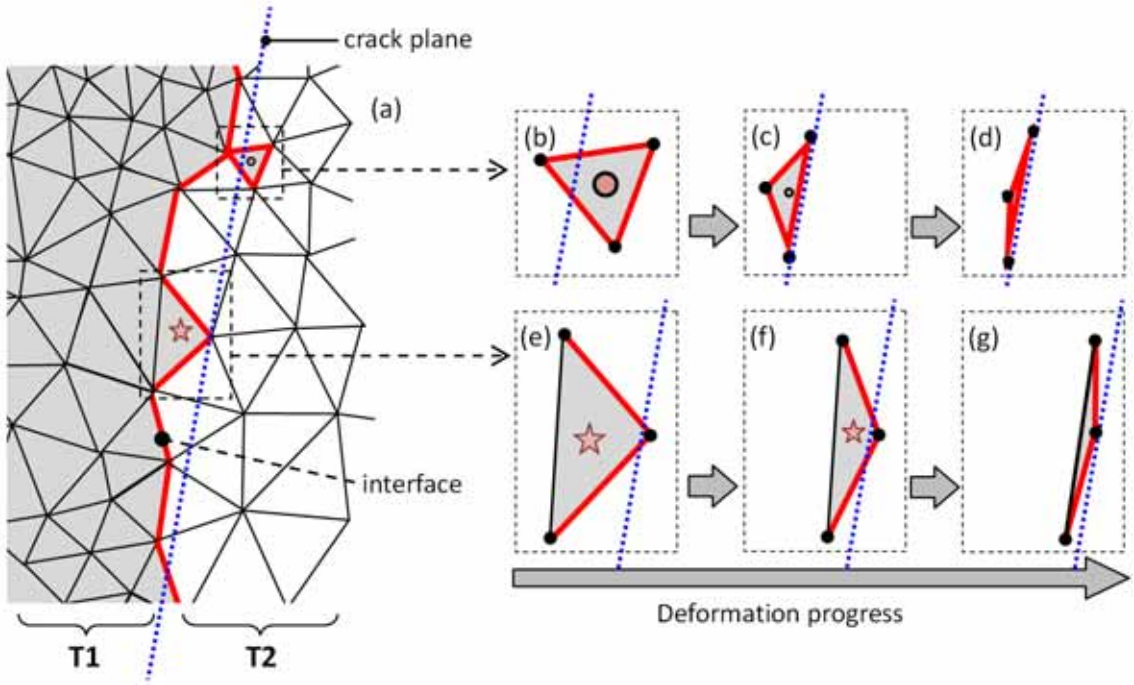


Figure 6.19: Examples of problem on the interface in 2D mesh

In case of a 3D mesh, the problematic tetrahedra may also become flat after deformation if these tetrahedra associate with more than 1 crack interface triangles. Figure 6.20 shows what will happen to these three problematic tetrahedra. On the upper line from left to right three tetrahedra associating with 2, 3 and 4 interface triangles are shown. The crack plane is supposed to be parallel to the

plane of the picture. Initially these tetrahedra are not flat which means that the dihedral angle θ is neither 0° nor 180°. The deformation to place the interface triangles onto the crack plane will make all of these three problematic tetrahedra flat as the one shown in the lower line of figure 6.20. The dihedral angle θ becomes 180°.

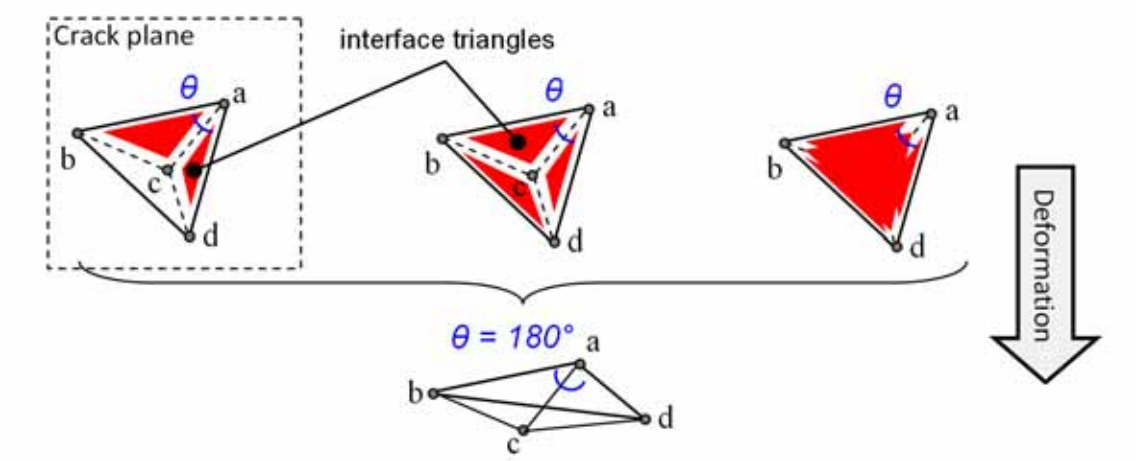


Figure 6.20: Examples of tetrahedra associating with 2, 3,or 4 crack interface triangles (upper) and their corresponding deformed versions (lower)

Therefore the pretreatment is performed so that on a 3D mesh (resp. 2D mesh), each tetrahedron (resp. triangle) of one set (either T_1 or T_2) share at maximum one triangle (resp. edge) with tetrahedron (resp. triangles) of the other set (either T_2 or T_1).

Therefore, tetrahedra which have 2 or 3 shared triangles (resp. triangles with 2 shared edges) should not take part to the definition of the CI since they will be flattened after the deformation. The processing of the set is performed as follows:

- In case of **2D** meshes, if a triangle in T_1 has **two** edges shared with triangles in T_2, it should be moved from moved from T_1 to T_2. In this way the third edge becomes an interface edge. An example of the problematic interface is shown in figure 6.21. The operated model is a cube in which two sub-parts T_1 and T_2 are identified (fig.6.21.b). Here, some problematic configurations can be encountered like the ones in figure 6.21.c. The yellow edges defined the interface subdividing the model into two sub-parts. Two triangles as-

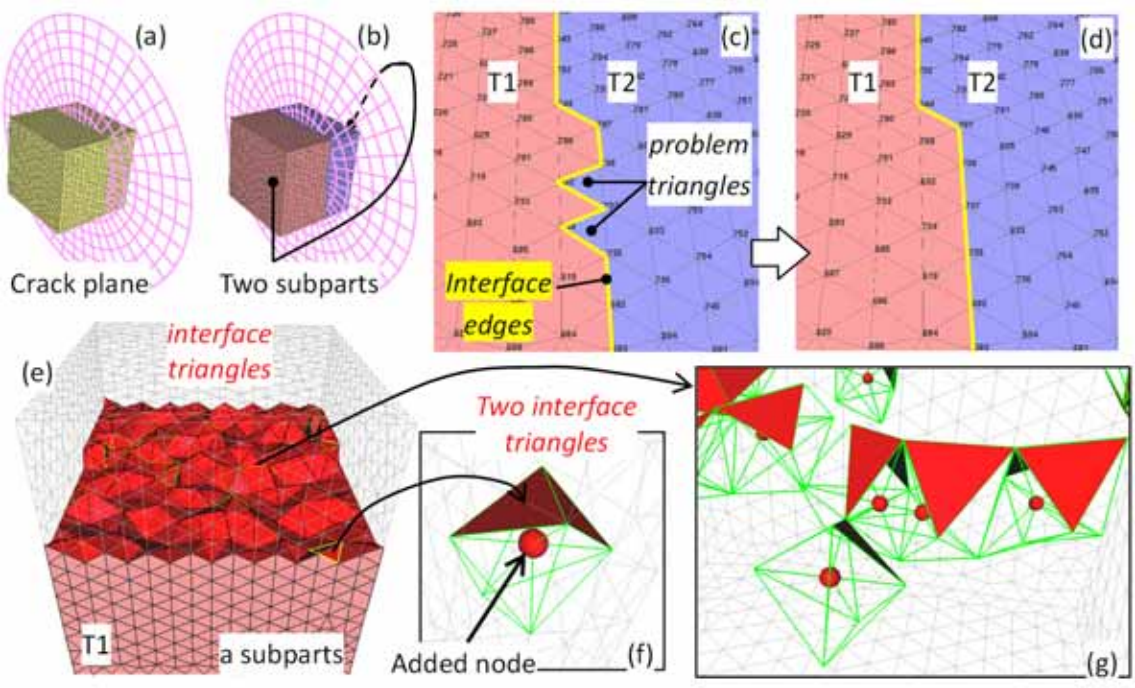


Figure 6.21: Examples of problem on the interface

sociated to two interface edges will be flatten once the interface edges are deformed onto the crack plane. Therefore the solution is to move those two triangles from the T_2 to T_1 and to replace the corresponding interface edges by the third edges of those two triangles (fig.6.21.d).

- In case of **2D** meshes, if a triangle in T_1 has **three** edges shared with triangles in T_2, it should be moved from the T_1 to T_2 and the associated edges should be removed from the interface set.

- In case of **3D** meshes, if a tetrahedron in T_1 has **two** triangles shared with elements in T_2, then the edge shared by the other two triangles should be split so that the tetrahedron is subdivided into two sub-tetrahedra which contain one of the two problematic triangle each. Figure 6.22.a shows the tetrahedron $abcd$ associated with two problematic triangles $\triangle acb$ and $\triangle adc$ and the dihedral angle θ that would become 180° after deformation. Here, the edge $b - d$ is split in two by inserting a new node o (fig.6.22.b). The tetrahedron is then subdivided into two tetrahedra $abco$ and $acdo$ and each

of them has one of the two interface triangles. All neighbour tetrahedra associated to the split edge $b-d$ should be also split. Figure 6.22.c shows all the neighbour tetrahedra in the original mesh and figure 6.22.d shows them split. If the model shown in figures 6.21.a and 6.21.b concerns a tetrahedral mesh, therefore a sub-part and the interface triangles are shown in figure 6.21.e. If the two interface triangles are deformed onto the crack plane the concerned tetrahedron will become flat. The proposed solution is to split the tetrahedron by inserting a new node. Figures 6.21.f and 6.21.g show different split tetrahedra.

- In the case of **3D** meshes, if a tetrahedron in T_1 contains **three** triangles shared with tetrahedra in T_2, this tetrahedron is moved from T_1 to T_2. In this way the fourth triangle becomes an interface triangle. This treatment is very similar to the case of 2D mesh where a triangle associates with 2 interface triangles.
- In case of **3D** meshes, if a tetrahedron in T_1 contains **four** triangles shared with tetrahedra in T_2, this tetrahedron is moved from T_1 to T_2 and the three associated edges are removed from the interface set.

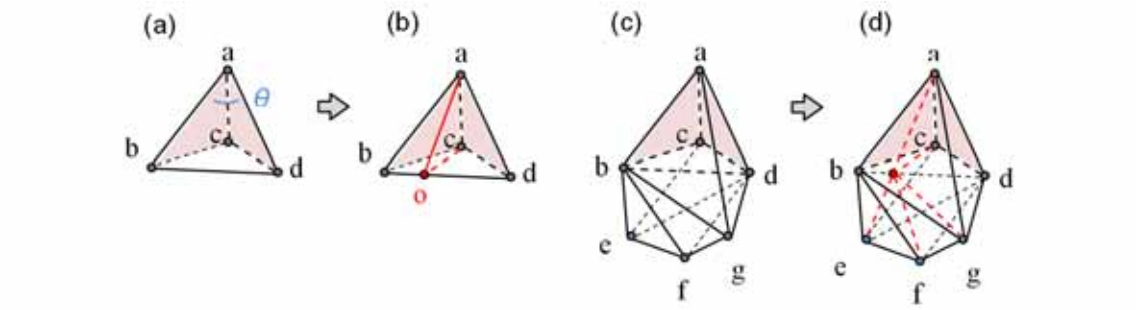


Figure 6.22: Splitting schema for the tetrahedron with 2 potential interface triangles

The algorithm 16 illustrates the crack interface pre-treatment in a tetrahedral mesh. The function "isOnBoundary (*tetrahedron*)" allows checking whether this *tetrahedron* associates with at least one boundary triangle which is shared exactly by one tetrahedron. The procedure "splitTetrahedraForCrackInterface

(*tetrahedron*)" will find all the concerned tetrahedra to split as shown in figure 6.22. The parameter *tetrahedron* play a role similar to the tetrahedron in figure 6.22.a and the concerns tetrahedra are equivalent to the ones in figure 6.22.c. The procedure "moveElementFromListToList (e, $L1$, $L2$)" allows moving the element e from its initial list ($L1$ or $L2$) to the other one ($L1$ or $L2$).

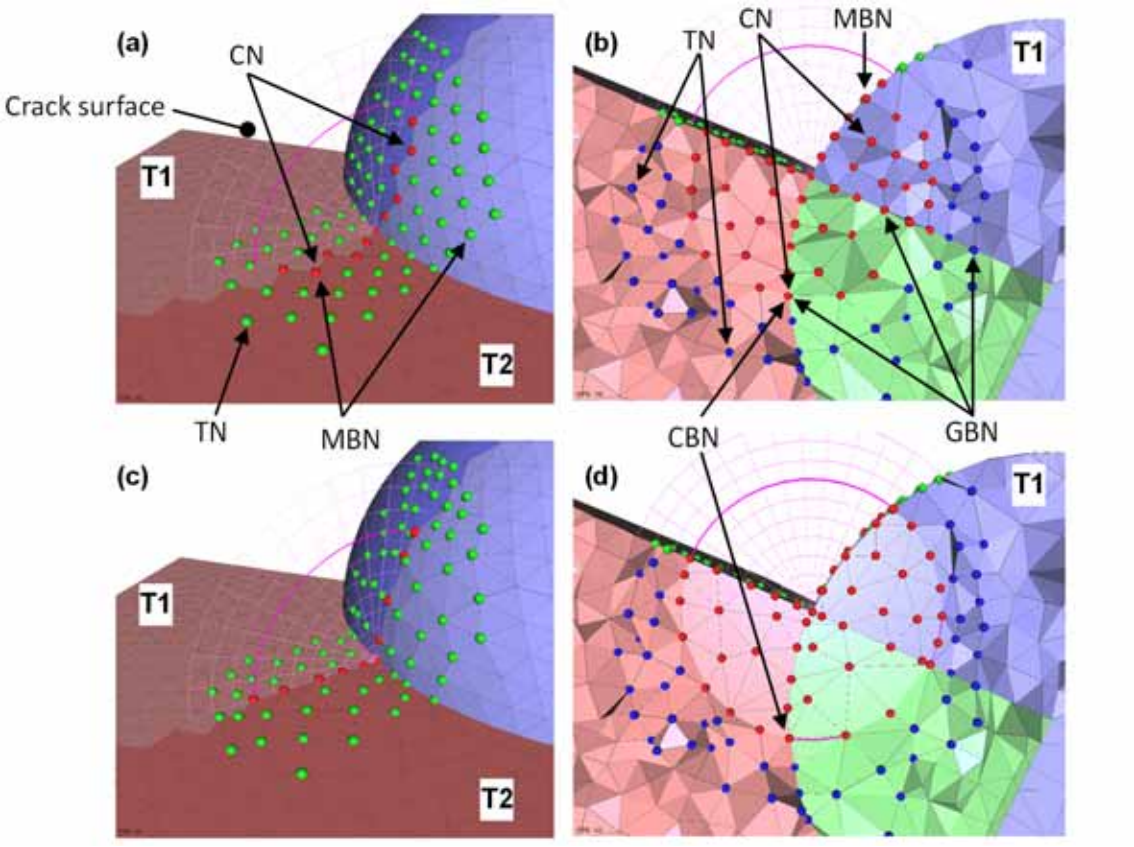


Figure 6.23: Constrained deformation for insertion of a planar crack into the 3D mesh containing 3 groups

6.3.4 Crack Interface deformation

Being the elements of the CI defined, the deformation process described in section 4.5 (p.129) is applied so that the elements of CI match the shape of the desired crack. It requires the identification of sets of elements to be moved in a different manner, i.e. using different constraints for each set:

Algorithm 16 Crack interface pre-treatment over a tetrahedral mesh

```
Procedure crackInterfacePretreatmentTetrahedral(crack As Crack)
 1: Variable flag As Boolean
 2: Variable n As Byte
 3: Variable tetrahedron As Tetrahedron
 4: Variable triangle, triangle', triangle'' As Triangle
 5: repeat
 6:   flag ← false
 7:   for all triangle in crack.CI do
 8:     for all tetrahedron associated with triangle do
 9:       n ← 0
10:       for all triangle' in tetrahedron do
11:         if isInList(triangle', crack.CI) then
12:           n ← n +1
13:         end if
14:       end for
15:       if n > 1 then
16:         flag ← true
17:         if n = 2 and isOnBoundary(tetrahedron) = false then
18:           splitTetrahedraForCrackInterface(tetrahedron)
19:         else
20:           moveElementFromListToList(tetrahedron, crack.T1, crack.T2)
21:           for all triangle'' in tetrahedron do
22:             if isInList(triangle'', crack.CI) then
23:               removeElementFromList(triangle'', crack.CI)
24:             else if isOnBoundary(triangle'') = false then
25:               addElementToList(triangle'', crack.CI)
26:             end if
27:           end for
28:         end if
29:       end if
30:       if flag = true then
31:         exit for
32:       end if
33:     end for
34:     if flag = true then
35:       exit for
36:     end if
37:   end for
38: until flag = true
End Procedure
```

1. the set CN of Crack Nodes constituted by all the nodes associated with the elements in CI are first identified. Figures 6.23.e and 6.23.f show the identified CN nodes in a 3D mesh.

2. to enable a smooth transition between the CI elements and the surroundings, a set of TN Transition Nodes is defined. It contains the nodes located in the i^{th} neighbourhood of the ones in CN. The bandwidth i can be user-defined, or it can be computed by dividing the biggest distance between one crack node and the crack plane by the mean edge length. The bigger it is, the smoother the transition will be, and the better the quality of the mesh will be. Figures 6.23.e and 6.23.f show some TN nodes identified with $i = 2$.

3. for a 3D mesh, all nodes belonging to only triangles shared by exactly two tetrahedra in a same set are internal, and all nodes associating with at least one triangle which is shared by exactly one tetrahedron are external. For a triangular mesh, all nodes belonging to only edges shared by exactly two triangles are internal, and all nodes associating to at least one edge which is shared by exactly one triangle are external.

4. the external nodes form a characteristics shape of the model. We use the tool presented in section 4.1 ($p_{.86}$) for detection of surface primitives such as plane, sphere, cylinder and so on in 2D/3D meshes. For the mesh shown in figure 6.23, different surface primitives (plane and sphere) are identified for external surfaces (fig.6.23.b). For preserving the shape of the original model we define also a set of MBN (Model Boundary Nodes). This set contains the CN and TN nodes on the mesh boundary (fig.6.23.e and 6.23.f).

5. since the crack operation should not influence the entire model, a deformation area has to be specified. The shape could be a rectangle, circle etc so that the CN nodes are delimited as well as the corresponding TN. The MBN needs to be updated also. We define a set of CBN (Crack Boundary Nodes) containing the CN nodes associating with delimited nodes and kept nodes on the crack plane. In case of figure 6.23 circular delimitation is

applied on the planar crack operation, some CBN nodes are shown (figure 6.23.f).

6. for preserving the group information the constraints from the VGB shape are also imposed during the deformation. A set of GBN (Group Boundary Nodes) is defined for all free nodes on the crack interface and in the transition zone.

Once the various sets of nodes are identified, geometric constraints can be assigned to drive the deformation process.

In case of 3D mesh, table 6.2 illustrates for nodes of different types (left column) the corresponding constraints (right column) that are applied.

Type of nodes	**Types of deformation constraint to apply**
CN	Stay on the crack plane
CBN	Stay on the delimitation shape
MBN	Stay on the surface of the external skin of the 3D mesh
GBN	Stay on the shape of corresponding group boundary
TN	Could move
Other nodes	Fixed

Table 6.2: Deformation constraints for crack operator on 3D mesh

In case of 2D mesh, table 6.3 illustrates for nodes of different types (left column) the corresponding constraints (right column) that are applied:

Type of nodes	**Types of deformation constraint to apply**
CN	Stay on the crack plane and on the mesh surface
CBN	Stay on the delimitation shape
MBN	Stay on the external curve boundary of the mesh
GBN	Stay on the shape of corresponding group boundary
TN	Stay on the mesh surface
Other nodes	Fixed

Table 6.3: Deformation constraints for crack operator on 2D mesh

In case of complex free form shapes, we use a tangent plane to the node to constrain locally the deformation (see section 4.5 on page 129 for more details).

During the mesh deformation, the deformation engine solves an under-constrained set of equations based on the mechanical model of a bar network coupled to the 2D / 3D mesh (section 4.5). The deformed model are shown in the figures 6.23.g and 6.23.h, they show separately the exterior and the interior of the 3D mesh.

Finally, to complete the insertion of the crack, the mesh entities belonging to the CI are duplicated to make a topological modification and the incident relations with edges, triangles and tetrahedra are accordingly updated.

6.3.5 Additional examples

The crack operation is applied on an academic example, a cube-like tetrahedral mesh (fig.6.24). The model to be cracked (fig.6.24.a) contains three tetrahedral groups that are separately shown in figures 6.24.b, 6.24.c and 6.24.d. Their VGBs are either planar or spherical or cylindrical. The crack process subdivides the model into two sub-parts T_1 and T_2. The sub-part T_1 is shown alone in figure 6.24.d before the deformation. After deformation the sub-part T_1 becomes the one shown in figure 6.24.e. The red nodes are the ones repositioned onto the crack plane. Table 6.5 gives some statistics on these results and is discussed hereafter.

The second experimentation of crack operator is performed on another academy example shown in figure 6.25. The model has 6 groups ($G1$ to $G6$) containing tetrahedra as shown in the figure 6.25.a. The crack plane is shown in purple and is limited by a circle that is drawn in bold. Figure 6.25.b shows the whole model where the crack interface is identified. The model is also subdivided into two sub-parts T_1 and T_2. The red nodes are on the crack interface and the green ones are the model skin nodes in the crack transition zone. The result of the deformation for the same view point is shown in figure 6.25.c. The internal views before and after deformation are shown in figures 6.25.d and 6.25.e. The sub-part is T_1 and the blue nodes are the ones in the transition zone.

The third example (fig.6.26) shows a crack operation performed on a tetrahedral mesh of a caisson, industrial model by EDF-R&D. Figure 6.26.a shows the 1/4 Caisson and the crack is performed solely on the stiffener part (fig.6.26.b). The crack interface identification as well as the two sub-parts T_1 and T_2 are shown in figure 6.26.c. Figure 6.26.e shows a zoom of the stiffener part before the defor-

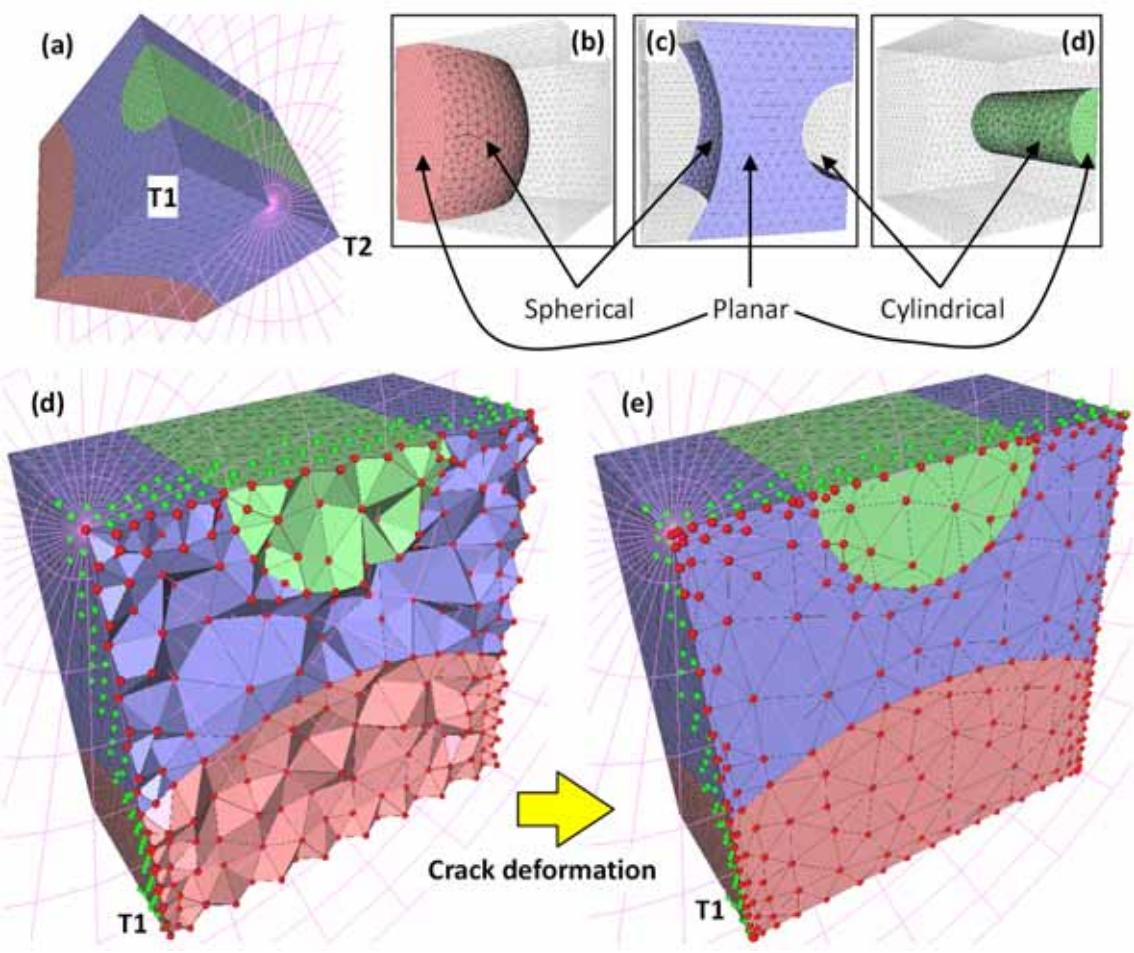


Figure 6.24: Crack operator applied on a cube-like tetrahedral mesh having 3 groups

mation whereas figure 6.26.f shows the same part after deformation. The crack is limited by a circle that is shown by bold circle on the crack plane. The triangles shared by one tetrahedron in T_1 and one in T_2 are composing the crack interface. Therefore, the red nodes are the crack interface nodes that are limited on the stiffener. The sub-part T_2 before and after the deformation is shown respectively in figures 6.26.g and 6.26.h. Since the sub-part T_1 is hided, the crack interface triangles are visible in the two pictures. One thing can be easily remarked that the crack deformation concerns only the stiffener and the rest of the Caisson is remained during the crack feature insertion.

The table 6.4 describes the number of various nodes used during the defor-

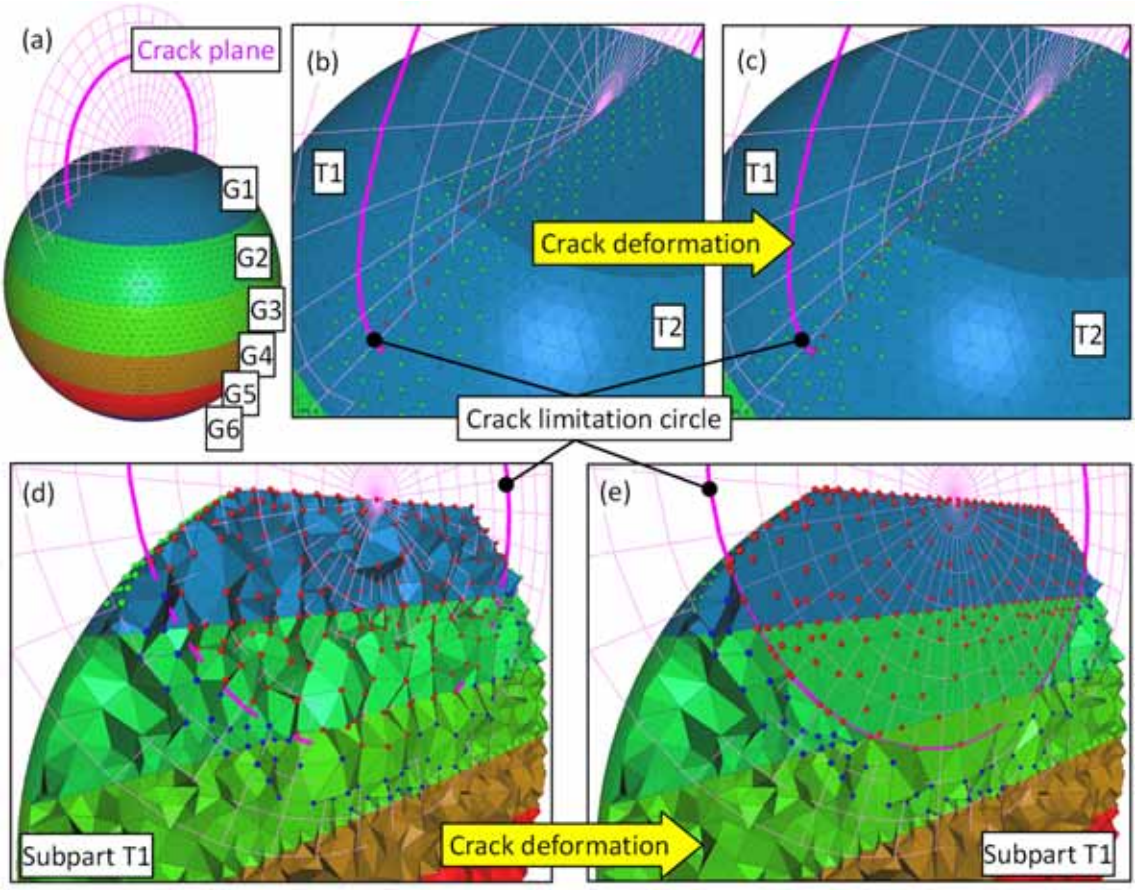


Figure 6.25: Crack operator applied on a sphere-like tetrahedral mesh having 6 groups

mation: the complete number of nodes in the mesh, the nodes (CN) moved onto the crack plane (some of them are model boundary nodes), the model boundary nodes (some of them are group boundary nodes) and the transition nodes (some of them are group boundary nodes).

For better understanding the quality of the operated mesh in comparison with the initial mesh is shown in the table 6.5. The computation of aspect ratio for tetrahedra is presented in the section 4.3 ($p._{120}$). The aspect ratio is between 0 and 1. Mean aspect ratio for all tetrahedra before and after merging is given in the table. The minimum and maximum of aspect ratio among all tetrahedra are also listed in the table.

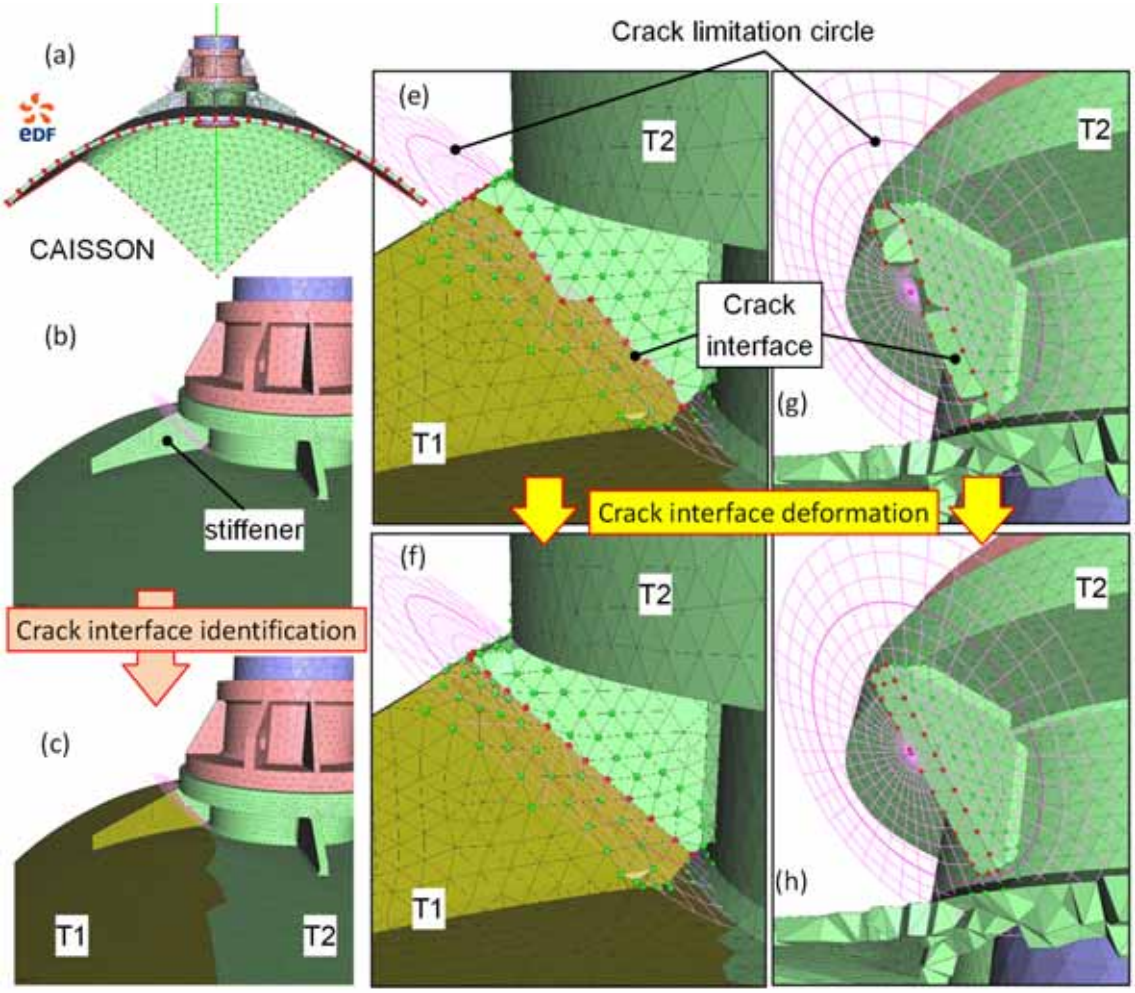


Figure 6.26: Insertion of a crack into a 3D mesh model (courtesy EDF-R&D)

According to the table of the quality resume it's evident that the prototyped operator needs to be improved in order to augment the produced model's quality. To this aim, we have already imagined a set of post-processing operations that would treat those specific configurations.

6.4 Mesh drilling operator

The mesh drilling operation introduced in [73, 74] consists in the insertion of a cylindrical through hole. The drilling cylinder is defined by three parameters:

Nodes \ Models	Box & Sphere fig.6.23	Cube fig.6.24	Spheres fig.6.25	Caisson 3D fig.6.26
Total	10471	5624	30119	22046
$CN + GBN$	47	237	233	29
$MBN + GBN$	92	471	276	165
$TN + GBN$	370	839	1209	27

Table 6.4: Various deformation nodes for cracked models

Aspect ratio Q		Box & Sphere fig.6.23	Cube fig.6.24	Spheres fig.6.25	Caisson 3D fig.6.26
Initial	Min	0.300	0.230	0.183	0.118
	Max	0.980	0.984	0.989	0.988
	Mean	0,736	0,687	0.705	0.673
Cracked	Min	0.000	0.000	0.000	0.008
	Max	0.980	0.976	0.989	0.988
	Mean	0,729	0,648	0.696	0.673

Table 6.5: Comparison of the aspect ratio Q for the various cracked models

a vector for the cylinder axis, a point on the axis and a radius of the cylinder. Schematically, the CAD-less mesh drilling operation to apply on semantically enriched meshes could be represented by figure 6.27. On this example, the model has two groups G1 and G2 and the drilling cylinder could be represented by a dash circle (fig.6.27.a). The mesh elements roughly enclosed in the drilling cylinder volume are removed (fig.6.27.b). The drilling interface is then computed and the drilling transition zone is defined (fig.6.27.c). The removal part is kept away from the mesh and the local deformation limited to the transition zone is applied to obtain a drilling interface that is cylindrical (fig.6.27.d).

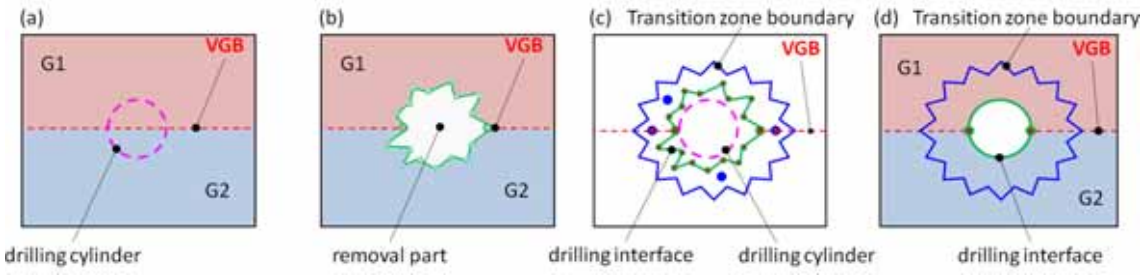


Figure 6.27: CAD-less mesh drilling operation schema

More concretely this operation is performed in several steps that are detailed hereafter (fig.6.28):

1. shape identification on the mesh boundary surface,
2. virtual group boundary computation on the existing groups,
3. rough drilling interface definition by computing the mesh elements roughly enclosed in the drilling cylinder volume,
4. drilling interface pretreatment to avoid surely flatten mesh elements during the deformation,
5. deformation transition zone definition around the drilling interface,
6. rough hole insertion by removing the elements of the enclosed part,
7. deformation constraints setting and interface deformation to match the cylindrical surface of the hole while smoothing the internal nodes positions,
8. semantics propagation because of the topology changes on the mesh.

6.4.1 Mesh elements classification

A drilling corresponds to a particular type of material removal operation. Roughly speaking it can be seen as a Boolean subtraction of a cylinder from the FE mesh (either 2D or 3D). The cylinder volume is implicitly defined by its enclosing tool surface. Therefore the first step of this operator requires the identification of the mesh elements to be removed so that the interface computed subsequently surrounds the whole cylindrical surface as much as possible.

With this aim, the mesh nodes are divided into two sets, I and O, which respectively indicate the nodes inside and outside the cylinder. Then, the mesh entities to be removed (RT) and to be kept (KT) are gathered. In the case of 3D mesh (resp. in case of 2D mesh), the set RT is defined as the set of all the tetrahedra (resp. triangles) having at least one node in the set labeled I. The remaining tetrahedra (resp. triangles) are put in the set KT. Note that in this

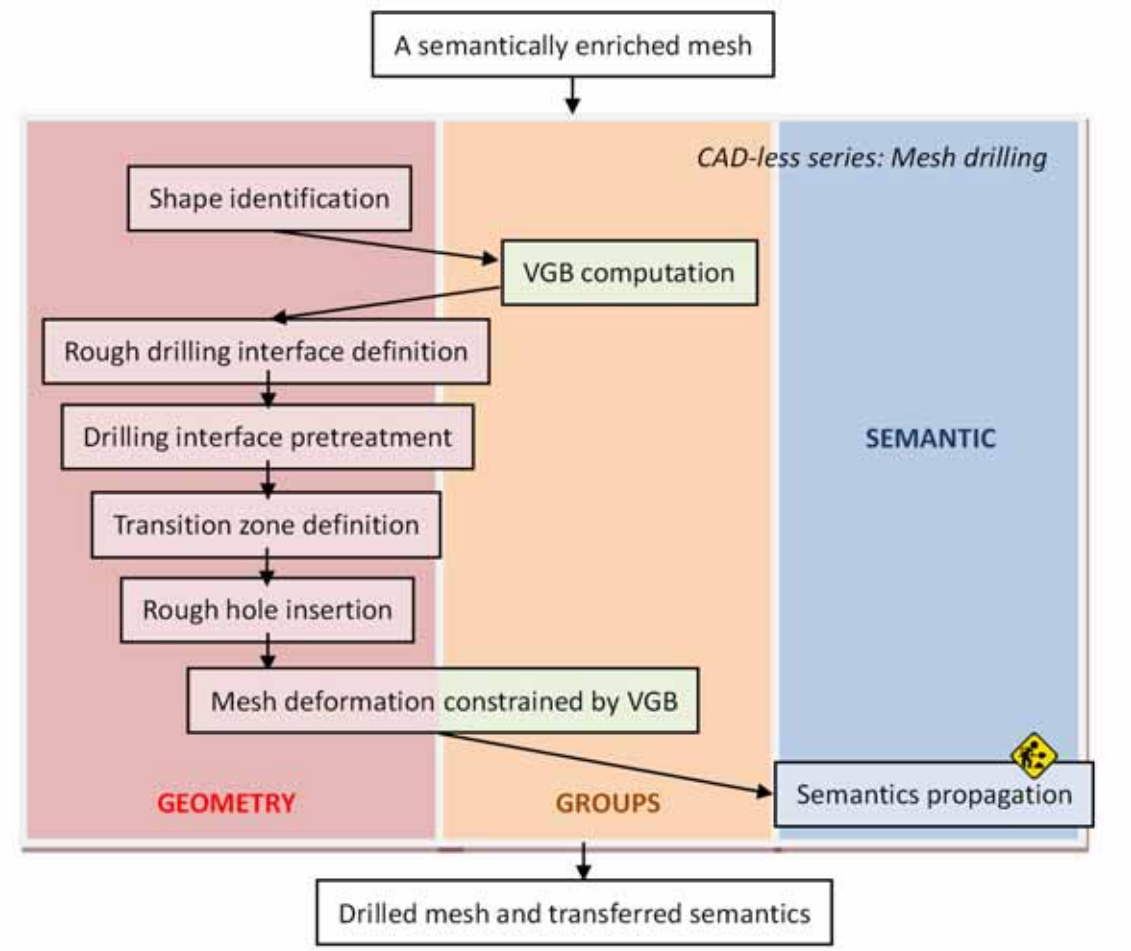


Figure 6.28: Workflow and necessary tools for mesh drilling operator

way, the mesh elements that are partially inside the cylinder are also defined to be removed. With this choice, the interface completely surrounds the cylinder, thus reducing the possible number of resulting "bad" quality, i.e. roughly flat, tetrahedra in the transition area after the deformation. Additionally, one can notice that, as a consequence of the shape of the target surface, i.e. the cylinder, the density of the nodes of the deformed interface will be higher than the one of detected interface, which means that some post-processing steps could also be useful here but have not been completely defined yet.

In figure 6.29 a tetrahedral mesh is used to test the drilling operation. The mesh has 3 tetrahedral groups ($G1$, $G2$ and $G3$) and the drilling cylinder is

specified as shown in figure 6.29.a. The mesh elements to be removed (RT) and the ones to be kept (KT) are then computed (fig.6.29.b). The removal part is shown solely in figure 6.29.c. The rough drilling interface can be then computed (fig.6.29.d).

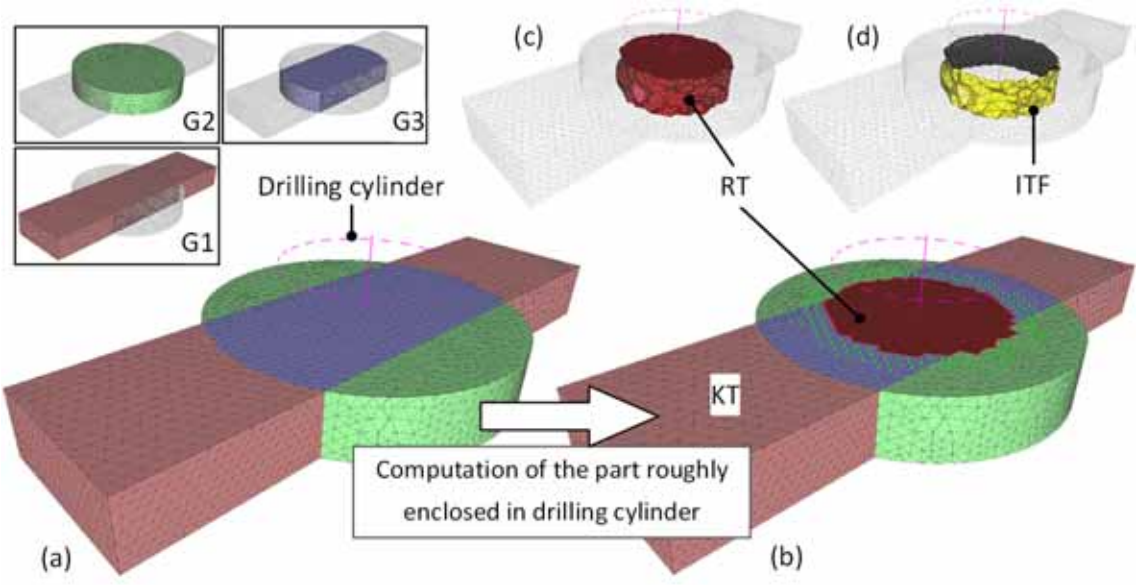


Figure 6.29: Mesh elements classification

6.4.2 Interface identification and pretreatment

Once the mesh elements to be removed are defined, they could not be removed immediately because original mesh elements are used to retrieve shape information necessary for guiding the deformation process. Nevertheless, the interface elements could be pre-computed. The hole interface is a set (ITF) of triangles for 3D mesh (resp. edges for 2D mesh) shared exactly by one removed and one kept mesh elements. For 3D mesh (resp. 2D), the set ITF is defined by all the triangles (resp. edges) which are shared by one tetrahedron (resp. triangle) in the set RT and one tetrahedron (resp. triangle) in the set KT.

A data-structure for the drilling operator is defined in algorithm 17.

In this data structure, the first three attributes are the parameters for defining the drilling cylinder. The fourth attribute is the concerned mesh to be drilled.

Algorithm 17 Data strucutre of Drill

Structure Drill

$point(3)$	As **Real**
$vector(3)$	As **Real**
$radius$	As **Real**
$mesh$	As **mesh**
RT	As **List**
KT	As **List**
ITF	As **List**

End Structure

The RT is a list storing mesh entities to be removed whereas the list KT contains the mesh entities to be kept. Therefore the list ITF consists in the drilling interface elements shared by one element stored in RT and one in KT. Using this data structure algorithm 18 of the procedure to define the drilling interface could be written by following.

To avoid bad behaviour during the successive deformation, it should be insured that in case of a 3D mesh (resp. 2D mesh) one tetrahedron (resp. one triangle) in KT is associated with only one triangle (resp. edge) in ITF. Tetrahedra in 3D mesh (resp. triangles in 2D mesh) which do not satisfy this condition will be flattened or flipped due to the deformation of the interface to match the cylinder.

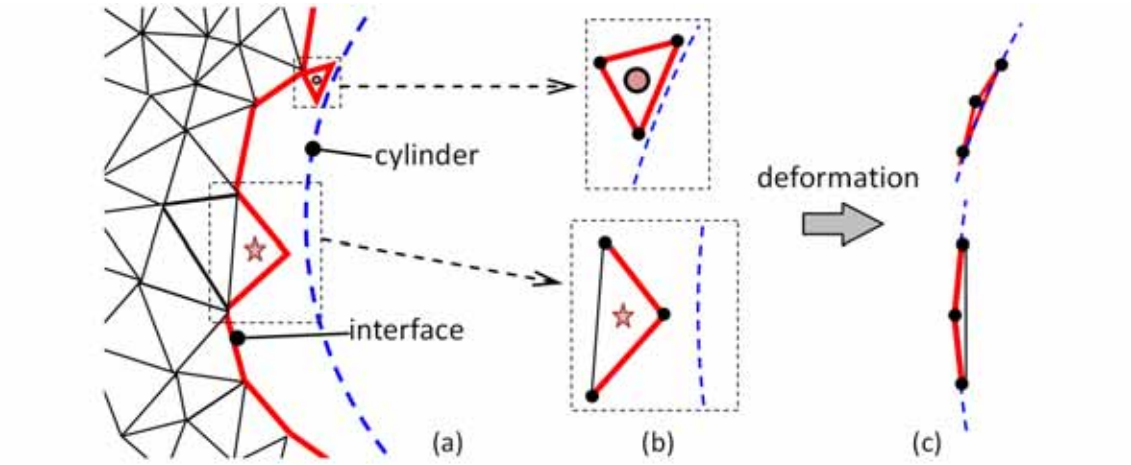


Figure 6.30: Two kept triangles associating with 2 or 3 drilling interface edges (a, b) and the corresponding deformed version (c)

Algorithm 18 Drilling interface identification over a tetrahedral mesh

```
Procedure drillingInterfaceIdentificationTetrahedral(drill As Drill)
 1: Variable vector(3) As Real
 2: Variable projectedPoint(3) As Real
 3: Variable distance As Real
 4: Variable n As Byte
 5: Variable list As List
 6: Variable t1,t2,tetrahedron As Tetrahedron
 7: Variable triangle As Triangle
 8: Variable node As Node
 9: for all node in drill.mesh do
10:    vector ← computeVectorByTwoPoints(drill.point, node)
11:    distance ← dotProductByTwoVectors(drill.normal, vector)
12:    projectedPoint(0) ← drill.point(0) + distance * drill.vector(0)
13:    projectedPoint(1) ← drill.point(1) + distance * drill.vector(1)
14:    projectedPoint(2) ← drill.point(2) + distance * drill.vector(2)
15:    if distanceByTwoPoints(node, projectedPoint) ≤ drill.radius then
16:       addElementToList(node, list)
17:    end if
18: end for
19: for all tetrahedron in drill.mesh do
20:    n ← 0
21:    for all node In tetrahedron do
22:       if isInList(node, list) then
23:          n ← n + 1
24:       end if
25:    end for
26:    if n > 4 then
27:       addElementToList(tetrahedron, drill.RT)
28:    else
29:       addElementToList(tetrahedron, drill.KT)
30:    end if
31: end for
32: for all triangle in drill.mesh do
33:    if triangle is shared by 2 tetrahedra then
34:       t1 ← first tetrahedron
35:       t2 ← second tetrahedron
36:       if isInList(t1, drill.RT) xor isInList(t2, drill.RT) then
37:          addElementToList(triangle, drill.ITF)
38:       end if
39:    end if
40: end for
End Procedure
```

To easily understand the reason, an example for the case of a 2D mesh is shown in figure 6.30. The blue dashed arc in figure 6.30.a represents the section of the cylinder so that the axis of the cylinder is perpendicular with the picture. Figure 6.30.a shows the elements of KT of the operated triangle mesh; for sake of clarity, the elements in RT are not drawn. The red edges constitute the interface. In this example, the triangle tagged by a pentagram has 2 edges belonging to the interface and the triangle tagged by a circle has 3 edges belonging to the interface. Configurations with triangles having 3 interface edges rarely happen except for the case of flipping triangles or numerical error. Figure 6.30.b shows a zoom of these two problematic configurations, and figure 6.30.c shows their possible shapes after the deformation of the interface. The three nodes of the triangles are on the circle, and the triangles are flattened and flipped. So, it is necessary to prevent such configurations using dedicated operations.

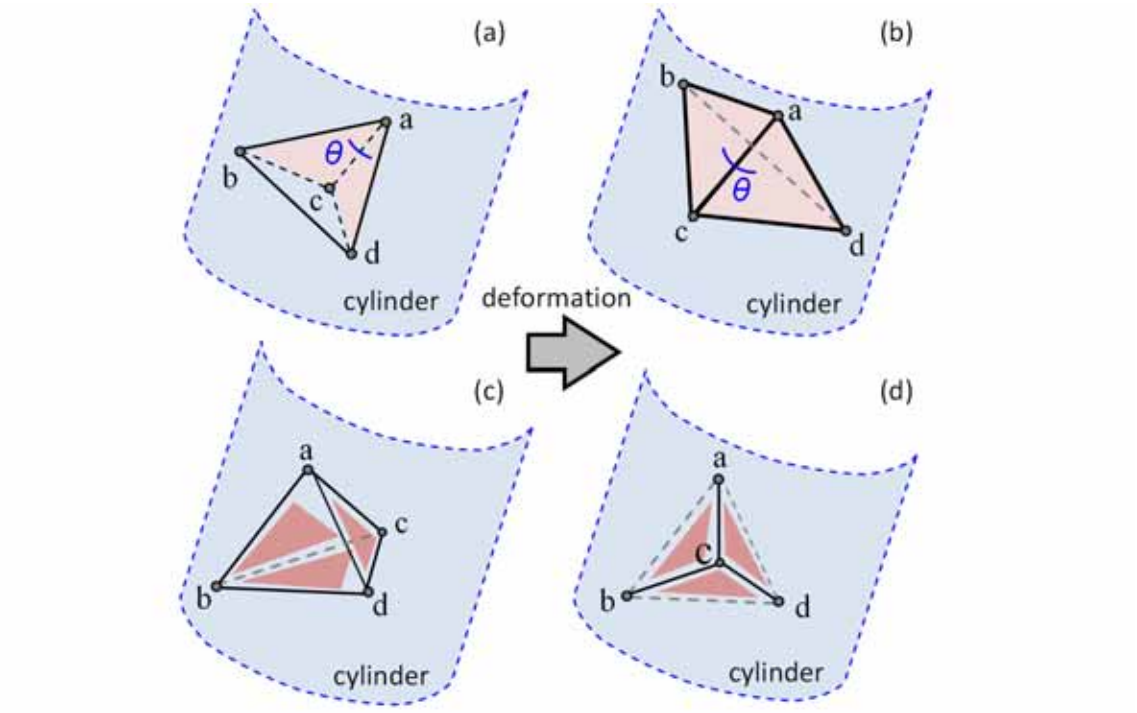


Figure 6.31: Examples of a kept tetrahedron associating with 2 drilling interface triangles (a) and with 3 drilling interface triangles (c) and their corresponding deformed versions (b, d)

Figure 6.31 illustrates a configuration where a tetrahedron is associated with

2 or 3 interface triangles. It could rarely happen that a tetrahedron is associated with 4 interface triangles. The tetrahedron *abcd* shown in figure 6.31.a is associated with two interface triangles $\triangle abc$ and $\triangle adc$. Figure 6.31.b presents the deformation result when all four nodes are on the cylinder (cutting tool). The dihedral angle between the two interface triangles is θ that is smaller than 180° before the deformation and bigger than 180° after the deformation. This tetrahedron is flattened and flipped. Figure 6.31.c corresponds to a case where the problematic tetrahedron associates with 3 interface triangles $\triangle abc$, $\triangle adc$ and $\triangle bcd$. Figure 6.31.d shows the result of the deformation. The node c is close to the triangle $\triangle abd$ and this node is at different sides of the triangle $\triangle abd$ before and after the deformation. This tetrahedron is also flattened and flipped. Similarly a tetrahedron associated with 4 interface triangles will be also flattened.

To prevent such configurations, in case of a 2D mesh, the solution is to remove the two (resp. three) concerned interface edges from the interface set ITF and add the third (resp. no) edge of the problematic triangle to ITF. At this stage, no mesh elements are removed, therefore while changing ITF we actually move this triangle from the KT to RT. Figure 6.32 shows how to apply the solution on the example illustrated in figure 6.30. The problematic triangles shown in figure 6.32.a are removed from the set KT (fig.6.32.b), the five initial interface edges on those two triangles are also removed from the interface and the third edge of the triangle tagged by a pentagram is added into the interface. The triangle tagged by a green heart symbol is the one associated with the newly added interface edge.

In case of a 3D mesh, the problematic tetrahedra associated to two or more interface triangles, are deleted, i.e. moved from the set KT to the set RT. Then the interface triangles are removed from the interface set, and are substituted by the other triangle(s) of the tetrahedron.

In case of a tetrahedron associated with two interface triangles, this approach is applied only when the remaining neighbour tetrahedra do not have again two interface triangles. In this case, which rarely occurs in the examples tested, the concerned tetrahedron is split such that it is substituted by two new tetrahedra having only one triangle in ITF. This is done by splitting the edge not shared by the two interface triangles and joining the new node to the other two non adjacent

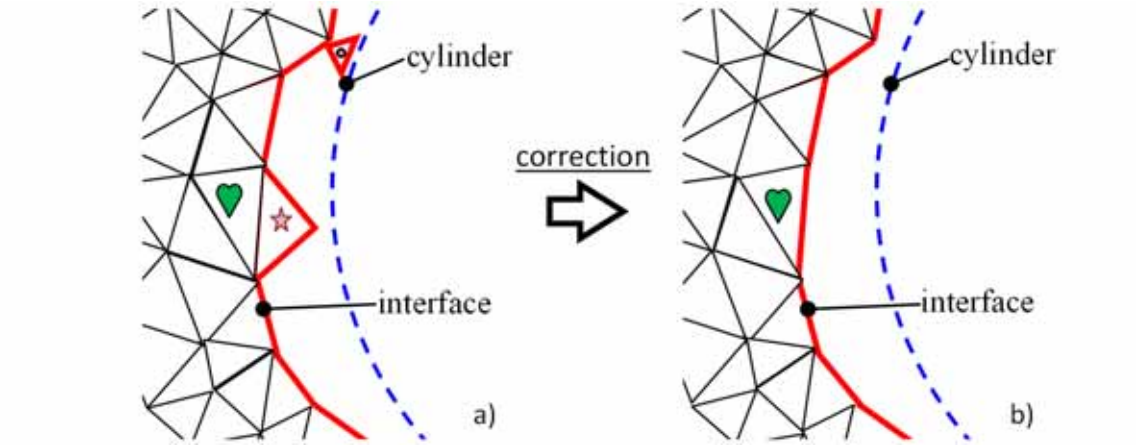


Figure 6.32: Drilling interface updating for case of kept triangle associating with 2 interface edges

vertices, thus a new triangle is obtained by considering these two new edges and the one shared by the interface triangles. The split scheme is exactly the same as the one used and shown in figure 6.22. A detailed algorithm for the drilling interface pre-treatment is shown (algo.alg:drillingInterfacePretreatmentTetrahedral) for a tetrahedral mesh.

6.4.3 Constraint definition and deformation

Once the elements of the interface set ITF are identified, the transition zone can be defined. All the nodes on the interface and the nodes in the transition zone will move for achieving the drilling surface taking into account mesh quality aspects. As previously said, the goal of the transition zone is to improve the mesh quality to make the variation of density from the interface to the unmodified mesh progressive. The transition zone nodes are the i^{th} neighbourhood of the ones associated with the ITF elements. The bandwidth i can be specified by the user or computed automatically. When automatically computed, its value is obtained by dividing the biggest distance between the interface nodes and the cylinder by the mean edge length. This gives an idea of how much the nodes have to move to achieve the target shape in relation with the density of the mesh. The bigger this value is the smoother the transition will be if we consider a larger neighbourhood, and the better the quality of the mesh will be.

Algorithm 19 Drilling interface pre-treatment over a tetrahedral mesh

```
Procedure crackInterfacePretreatmentTetrahedral(dill As Drill)
 1: Variable flag As Boolean
 2: Variable n As Byte
 3: Variable tetrahedron As Tetrahedron
 4: Variable triangle, triangle′, triangle″ As Triangle
 5: repeat
 6:   flag ← false
 7:   for all triangle in dill.ITF do
 8:     for all tetrahedron associated with triangle do
 9:       if isInList(tetrahedron, drill.KT) then
10:         n ← 0
11:         for all triangle′ in tetrahedron do
12:           if isInList(triangle′, drill.KT) then
13:             n ← n +1
14:           end if
15:         end for
16:         if n > 1 then
17:           flag ← true
18:           moveElementFromListToList(tetrahedron, drill.KT, drill.RT)
19:           for all triangle″ in tetrahedron do
20:             if isInList(triangle″, drill.ITF) then
21:               removeElementFromList(triangle″, drill.ITF)
22:             else if isOnBoundary(triangle″) = false then
23:               addElementToList(triangle″, drill.CI)
24:             end if
25:           end for
26:         end if
27:       end if
28:       if flag = true then
29:         exit for
30:       end if
31:     end for
32:     if flag = true then
33:       exit for
34:     end if
35:   end for
36: until flag = true
End Procedure
```

To assign the various constraints to the interface and transition nodes, the different shape information indicated in subsection 4.5.4 (p.138) needs to be derived as well as the classification of the concerned nodes.

At first, the mesh boundary elements of the transition area are detected. For 3D mesh, the boundary is the connected set of triangles that associate only with one tetrahedron of the mesh. Similarly, for 2D mesh, the boundary is the connected set of edges that associate with only one triangle and the "body" is the set of all the triangles in the mesh. In the case of 2D mesh not only the curve shape of the boundary but also the surface shape of the mesh body is taken into account during the deformation. All this shape information is computed with all original mesh entities before the deletion of the RT elements. Then, the shape of the elementary groups is computed for all those present in the original FE mesh affecting elements in the interface and transition area. Once all important shape information is computed, the mesh elements in the set RT are removed from the mesh. Finally, the nodes on which to assign the constraints are identified and classified.

Similarly to the crack insertion problem, the geometric constraints specification will not be the same for all the mesh entities, and an identification of specific node sets to be constrained is required.

First, a set HN (Hole Nodes) gathers together all the nodes associated with ITF elements. Second, a set TN (Transition Nodes) is defined similarly to the ones used in the crack insertion problem, i.e. using a bandwidth of i^{th} neighbourhood.

For preserving the shape of the original model we define also a set of MBN (Model Boundary Nodes). This set contains the HN and TN nodes on the mesh boundary.

For preserving the group information the constraint from the virtual group boundary shape is also imposed during the deformation. One important point is that the VGB should be computed before the deletion of the elements in RT. A set of GBN (Group Boundary Nodes) is defined for all free nodes on the crack interface and in the transition zone.

The elements corresponding to the hole, RT tetrahedra (resp. RT triangles) are then removed from the mesh model. Then, geometric constraints are assigned

to different nodes depending on the sets to which they belong.

In case of 3D mesh, table 6.6 illustrates for nodes of different types (left column) the corresponding constraints (right column) that are applied.

Type of nodes	**Types of deformation constraint to apply**
HN	Stay on the drilling cylinder
MBN	Stay on the surface of the external skin of the 3D mesh
GBN	Stay on the shape of corresponding group boundary
TN	Could move
Other nodes	Fixed

Table 6.6: Deformation constraints for drilling operator on 3D mesh

In case of 2D mesh, table 6.7 illustrates for nodes of different types (left column) the corresponding constraints (right column) that are applied.

Type of nodes	**Type of deformation constraints to apply**
HN	Stay on the drilling cylinder and on the mesh surface
MBN	Stay on the external curve boundary of the mesh
GBN	Stay on the shape of corresponding group boundary
TN	Stay on the mesh surface
Other nodes	Fixed

Table 6.7: Deformation constraints for drilling operator on 2D mesh

The constraints being defined, the deformation engine detailed in section 4.5 (p.129) gives a solution in term of local deformation of the drilled hole.

Figure 6.33 shows the rough hole generation and the cylindrical hole deformation based on the example presented in figure 6.29. Figures 6.33.a and 6.33.c show from different view point the model after the deletion of the tetrahedra (RT) enclosed in the cylinder volume. After deletion the hole interface appears and the hole nodes (HN) can be seen. Figures 6.33.b and 6.33.d show the model after the deformation. The transition zone is defined with two rings of neighbours. The group boundary nodes (GBN) in the transition zone are identified. The shapes of the group boundaries as well as the shapes of the model skins are preserved.

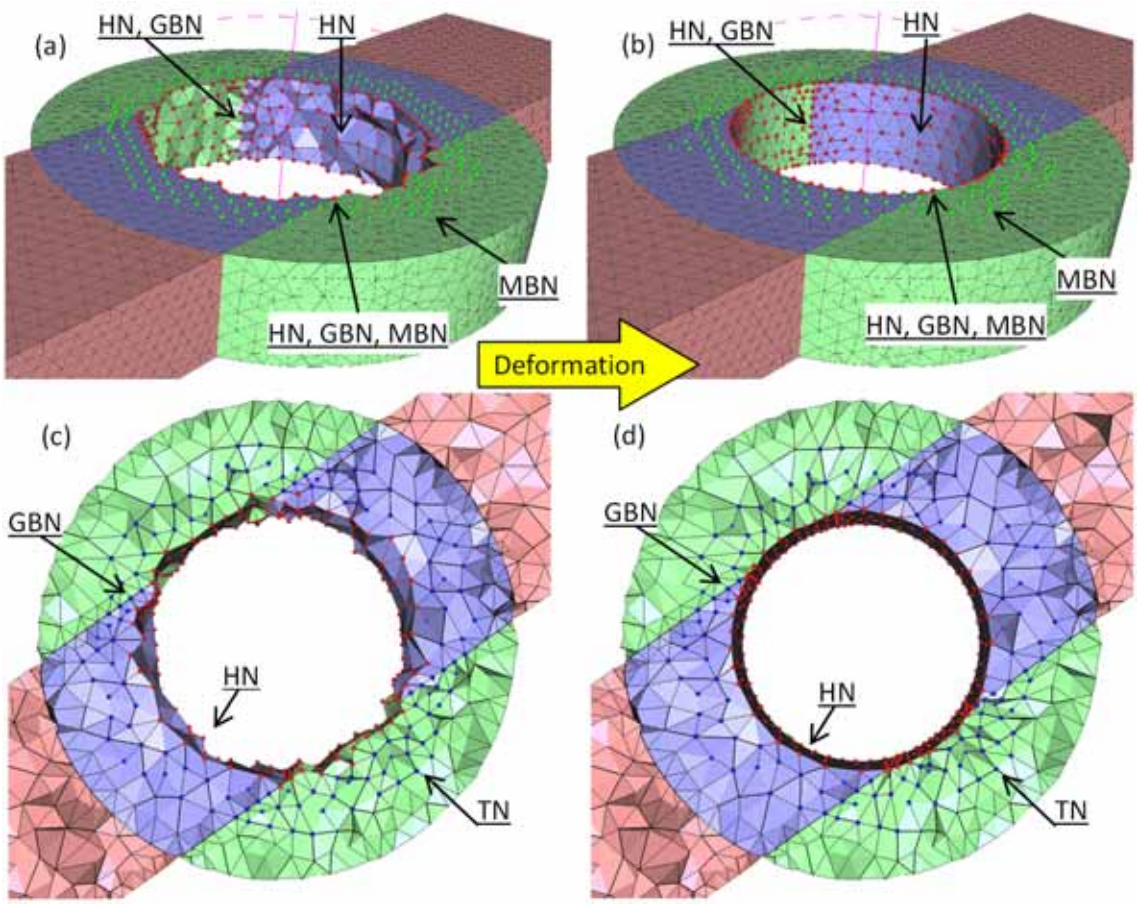


Figure 6.33: Rough hole generation (a,c) and mesh deformation (b,d)

6.4.4 Additional examples

Figure 6.34 illustrates the whole process on a cube-like tetrahedral mesh that has already been used to apply the crack operation (fig.6.24). Three groups of tetrahedra as well as their VGB are shown in figure 6.34.b. The tetrahedra enclosed in the drilling cylinder volume are removed (fig.6.34.c). Different nodes are identified in order to specify different constraints as described in the subsection 6.4.3. The result of the deformation is shown in figure 6.34.d. Here again, the shapes of the group boundaries as well as the shapes of the model skins are preserved. Table 6.5 (p.222) gives some numerical values for the aspect ratio Q.

The drilling operator has also been applied several times on the Stanford Bunny on which 4 groups have been associated so that the resulting VGBs are

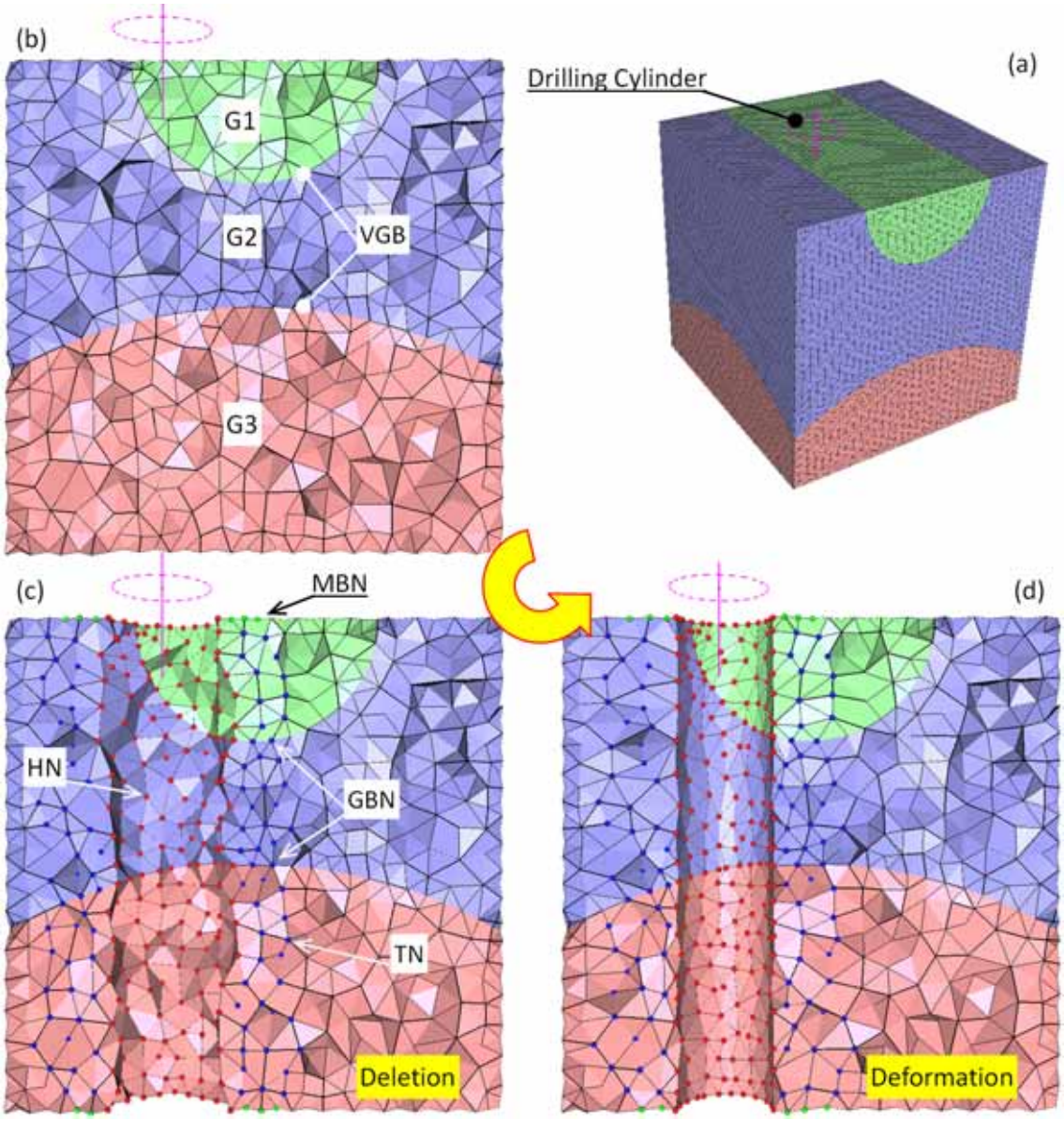


Figure 6.34: Creation of a through hole in a 3D mesh while preserving the shapes of the groups

spherical (fig.6.35). Figure 6.35.a shows the entire bunny on which several drills are indicated. Four tetrahedral groups are created for representing respectively different parts: Right Hear, Left Hear, Head and Body. The VGBs shared by two groups (i.e. not on the model boundary) are spherical. Figures 6.35.b, 6.35.c and 6.35.d show separately zoomed view of three holes and figure 6.35.e shows an internal view of the one shown in figure 6.35.d. Different categories of nodes are indicated in the figure. Here again, the resulting shapes satisfy the constraints arising from the shape of the tool, the shape of the VGB as well as the shape of the outer skin of the Bunny, thus preserving the associated semantics.

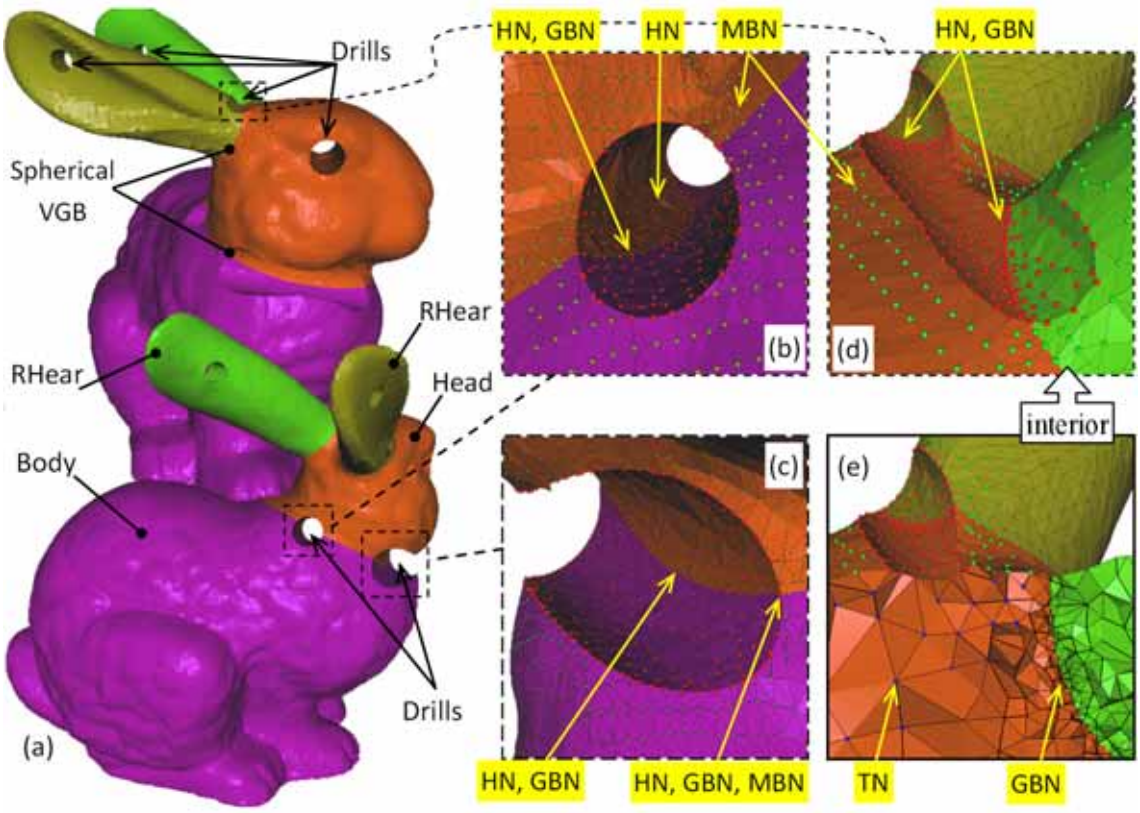


Figure 6.35: Multiple drills on the Stanford Bunny characterized by four groups of tetrahedra having spherical VGBs

The last example (fig.6.36) corresponds to the hole making operation performed on the tetrahedral mesh of the caisson model introduce in the previous subsection (fig.6.36). This operation is performed on the stiffener close to the

Nodes \ Models	Box & Cylinder fig.6.33	Cube fig.6.34	Bunny fig.6.35	Caisson 3D fig.6.36
Total	7761	15661	91872	22046
$HN + GBN$	362	388	1832	19
$MBN + GBN$	452	288	1763	46
$TN + GBN$	692	1316	5014	21

Table 6.8: Various deformation nodes for drilled models

cylinder part (fig.6.36.a). Figure 6.36.b shows a zoom of the zone surrounded by a rectangle. Different mesh entity sets are identified to enable the specification of constraints required for the deformation of the 3D elements located in the area of the given stiffener.

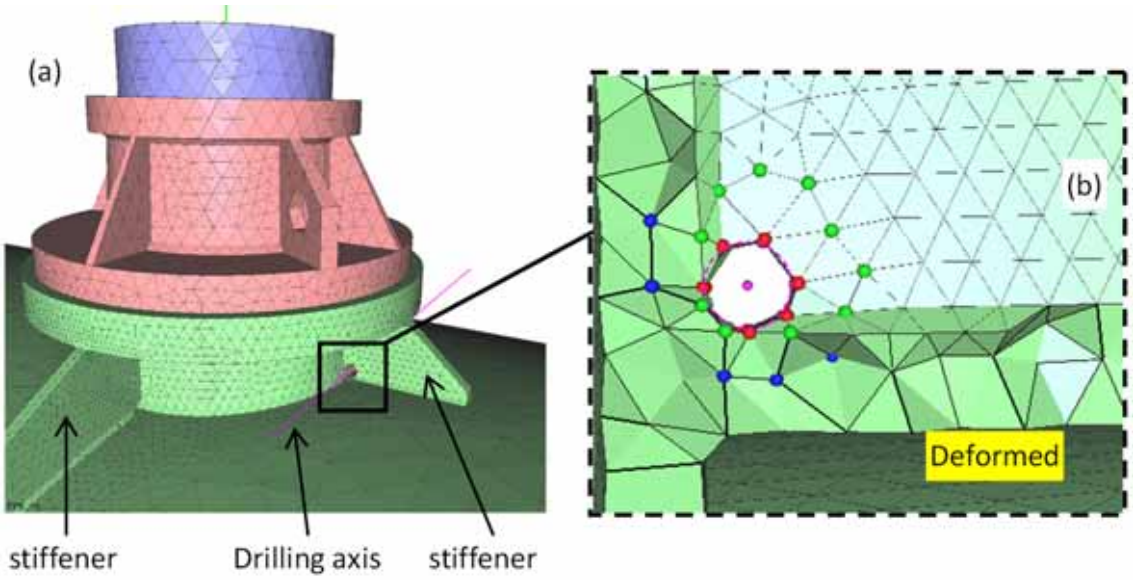


Figure 6.36: Insertion of a cylindrical hole into the 3D mesh of a caisson model (courtesy EDF-R&D)

The table 6.8 describes the quantity of various nodes used during the deformation: the overall number of nodes in the mesh before drilling, nodes (HN) moved onto the hole cylinder (some of them are group boundary nodes), model boundary nodes (some of them are group boundary nodes) and transition nodes (some of them are group boundary nodes).

To better understand the quality of the operated mesh in compared with the

Aspect ratio Q		Box & Cylinder fig.6.33	Cube fig.6.34	Bunny fig.6.35	Caisson 3D fig.6.36
Initial	Min	0.251	0.230	0.028	0.118
	Max	0.980	0.992	0.982	0.988
	Mean	0,695	0,709	0.663	0.673
~~Cracked~~ Drilled	Min	0.007	0.028	0.002	0.152
	Max	0.980	0.992	0.982	0.988
	Mean	0,672	0,699	0.656	0.671

Table 6.9: Comparison of the aspect ratio Q for the various drilled models

initial mesh, the table 6.5 summarises the quality evolution. The computation of the aspect ratio for the tetrahedra is presented in the section 4.3 (p.120). The aspect ratio evolves between 0 and 1. Mean aspect ratio for all tetrahedra before and after merging is given in the table. The minimum and maximum of aspect ratio among all tetrahedra are also listed in the table.

According to the table of the quality, it is clear that the prototyped operator needs to be improved in order to augment the produced model's quality. To this aim, several post-processing steps have already been imagined but are not part of this manuscript.

6.5 Mesh filleting operator

The last prototyped CAD-less mesh modelling operator is the so-called sharp edge filleting. It consists in converting the sharp features of the model into a smooth/rounded area. It is very important for obtaining good results during the FEA since the stress concentrates much more in the sharp areas than in the smooth areas. Therefore, a very common way to reduce the risk of having too much stress consists in rounding a bit the sharp edges.

6.5.1 Overview of the mesh filleting process

The prototyped mesh filleting operator deforms locally the mesh by moving nodes surrounding the sharp edges. The main steps are listed below (fig.6.37):

1. all sharp edges are identified,

2. the VGBs are computed,
3. the relative sharp edges are identified based on user-specified sharp reference edges,
4. the filleting areas are defined under the notion of neighbour range,
5. the surface mesh or boundary of volume mesh are deformed under constraints of VGBs,
6. the volume mesh interior are relaxed under constraints of VGBs.

The step 1) for computing all sharp edges on a mesh has already been presented in the section 4.2 (p.109). The step 2) for computing the virtual group boundary (VGB) has been detailed in the section 5.2 (p.152). The step 3) consists in finding relative sharp edges based on the ones selected by the user. The step 4) defines the filleting area whose size will implicitly define the filleting radius. The step 5) deforms locally in the filleting zone the surface mesh or the boundary (hull) of volume mesh in order to replace the sharp corner by a smooth band. The step 6) concerns only the volume mesh and relaxes the internal mesh elements by fixing the deformed boundary (hull) surface mesh. In the following subsections the steps 3) to 6) are presented more in details.

6.5.2 Definition of sharp edges to be filleted

This subsection presents how the sharp edges are selected based on the user-reference sharp edges. A foregoing step has to identify all sharp edges and this step will find a subset of these sharp edges according to the reference edge imposed by user. It is supposed that the foregoing step of all sharp edge computation is affirmed by user and he/she will impose one or several sharp edges already detected by the system. In case the sharp edge that user wants to impose is not identified by the system, the user has to change sharp edge detection parameters in order to re-compute the sharp edges (section 4.2 on page 109).

When the reference edge is given, the searching of relative sharp edges is deployed by walking along the neighbour sharp edges. That's to say from each

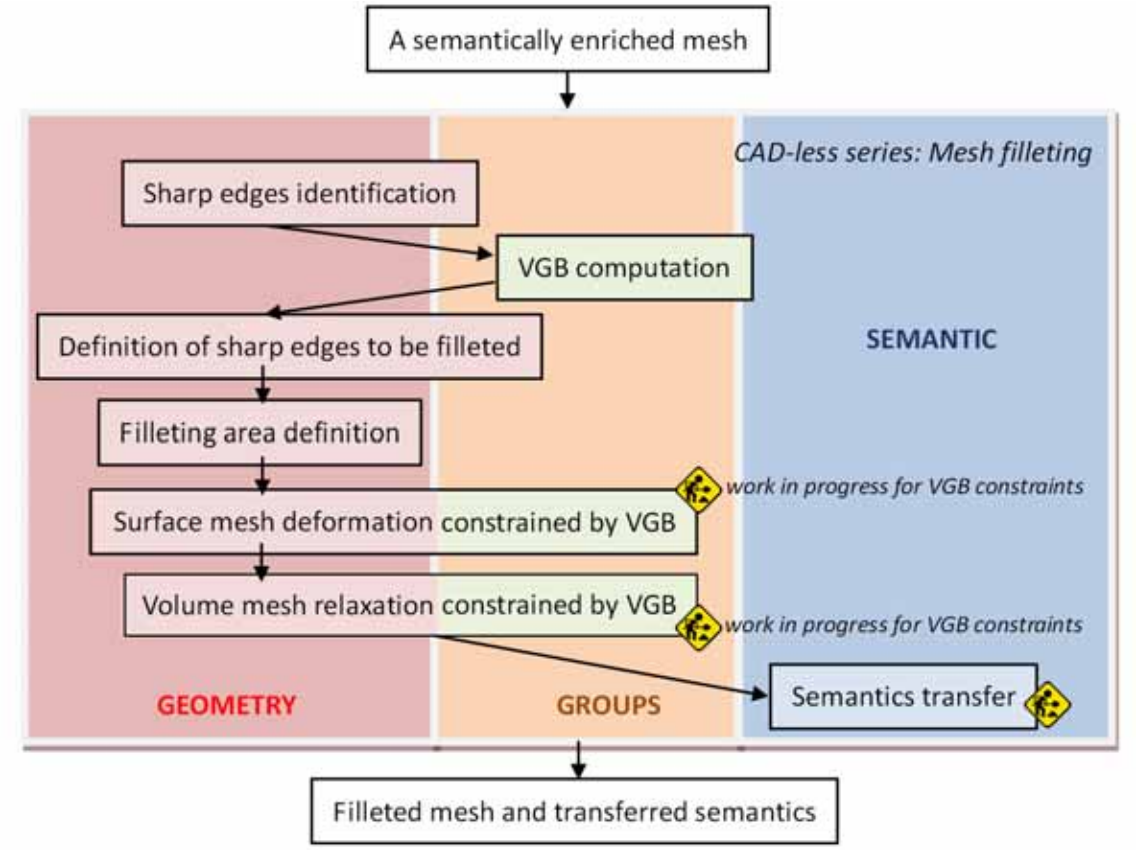


Figure 6.37: Workflow and necessary tools for the mesh filleting operator

of two extremities the associated sharp edge(s) is found and the other extremity of the found edge is used to find other sharp edges. The searching process runs recursively/iteratively and the condition to stop the process can be of three different cases:

(1) Leaving from one of the two extremities of the reference edges, **the searching process meet a node associating with only one sharp edge**. In this case the searching process stops at this node call "end point". Figure 6.38.a shows a parallelepiped on which some sharp corners are already rounded. Therefore the all sharp edges computed are shown in figure 6.38.b. If the user indicates a reference edge as shown in figure 6.38.c the relative sharp edge searching process from each extremity will stop at the two end-points. Both of the two end-points associate with only one sharp edge that is selected as relative sharp edges.

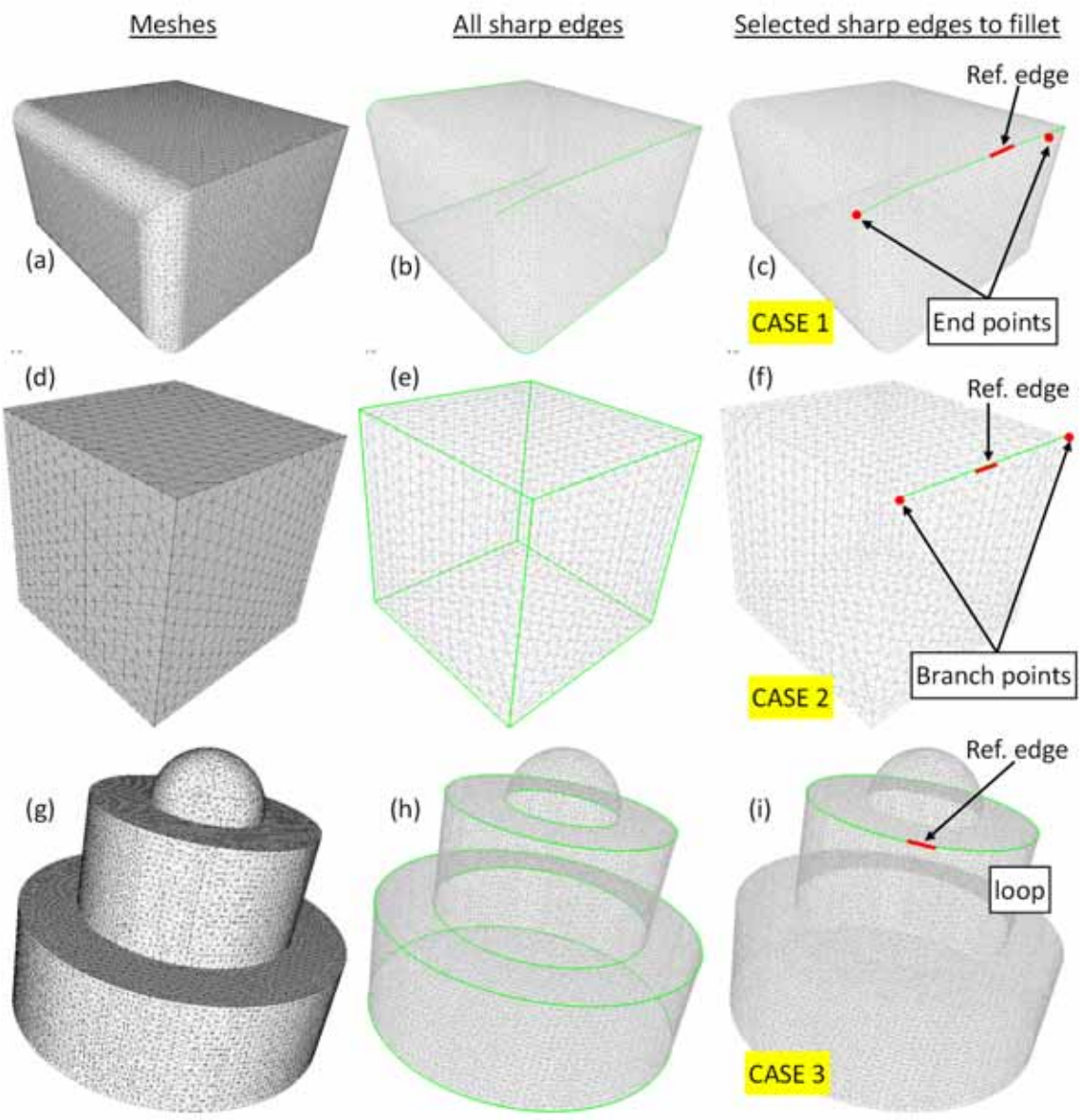


Figure 6.38: Relative sharp edges searching process stops in 3 cases

(2) Leaving from one of the two extremities of the reference edges, **the searching process meet a node associating with more than two sharp edges**. In this case the searching process stops at this node call "branch node". Figure 6.38.d shows a cube-like mesh and all sharp edges computed are shown in figure 6.38.e. If the user indicates a reference edge as shown in figure 6.38.f the relative sharp edge searching process from each extremity will stop at the two branch points. Both of the two end-points associate with three sharp edges and only one of the three sharp edges is selected to fillet.

(3) Leaving from one of the two extremities of the reference edge, **the searching process meets the second extremity of the reference edge**. In this case the searching process stops at this second extremity of the reference edge. The other searching process leaving from the second extremity of the reference edge is canceled. The selected relative sharp edges form a loop. Figure 6.38.g shows a mesh model on which all sharp edges are computed (fig.6.38.b). If the user indicates a reference edge as shown in figure 6.38.i the selected relative edges will be a loop.

The sharp edges to fillet selection could be although a mixture of these cases but for each searching process leaving from an extremity of reference edge will belong to the three presented cases. The algorithm 20 for the filleting sharp edges selection is detailed hereafter. In this algorithm, the input $sharpEdges$ is the list containing all pre-computed sharp edges on a mesh model. The $refEdge$ is the user indicated sharp edge and the output variable $selectedEdges$ will contain all found relative sharp edges to fillet. The variable $searchStopMode$ corresponds to the three stop cases presented in the previous paragraph (fig.6.38).

6.5.3 Filleting area definition

Once the sharp edges to be filleted are selected, the filleting area is defined around these edges. The filleting area is defined under the notion of neighbourhood previously presented. The size of this filleting area represents implicitly the filleting radius, which means that bigger the filleting area is bigger the filleting radius is. So, in the prototyped filleting operator, the radius of the fillet is not imposed directly even if it would not be difficult to extend this operator by taking into

Algorithm 20 Sharp edge selection

```
Function sharpEdgeSelection(sharpEdges As List, refEdge As Edge) As List
 1: Variable n, i As Byte
 2: Variable selectedEdges As List
 3: Variable curEdge, edge As Edge
 4: Variable curNode As Node
 5: Variable searchStopMode As Byte
 6: addElementToList(refEdge, selectedEdges)
 7: for i = 0 to 1 do
 8:   curEdge ← refEdge
 9:   curNode ← refEdge.nodes(i)
10:   repeat
11:     n ← 0
12:     for all edge in curNode.edges do
13:       if isInList(edge, sharpEdge) then
14:         n ← n +1
15:         if edge ≠ curEdge then
16:           curEdge ← edge
17:         end if
18:       end if
19:       if n=1 then
20:         searchStopMode ← 1
21:       else if n > 2 then
22:         searchStopMode ← 2
23:       else if n = 2 then
24:         if curEdge = refEdge then
25:           searchStopMode ← 3
26:         else
27:           addElementToList(curEdge, selectedEdges)
28:           if curEdge.nodes(0) ≠ curNode then
29:             curNode ← curEdge.nodes(0)
30:           else
31:             curNode ← curEdge.nodes(1)
32:           end if
33:         end if
34:       end if
35:     end for
36:   until searchStopMode = 0
37:   if searchStopMode = 3 then
38:     exit for
39:   end if
40: end for
41: return selectedEdges
End Function
```

account the specific filleting radius. Note that for a volume mesh the filleting area is defined on the neighbourhood **with nodes on the external skin and also the nodes inside the model**.

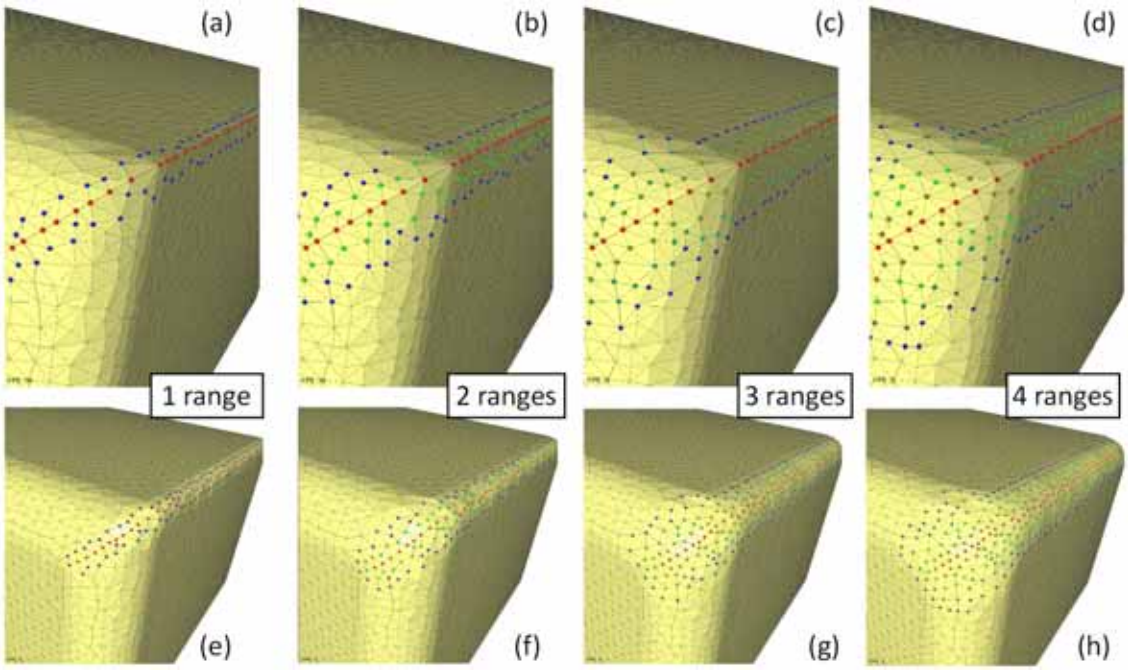


Figure 6.39: Different filleted areas based on various range numbers (from 1 to 4)

Figure 6.39 shows the experimentation of different sizes for the filleted area. It consists in the partially filleted cube mesh model and the selected sharp edges to fillet that are already shown in figure 6.39.c. Here, for defining the filleted zone, different ranges of neighbourhood around the selected sharp edges are used (from 1 to 4 neighbours). The red nodes are the ones on the selected sharp edges to fillet and the other nodes in different colors are the ones in the neighbourhood. Figures 6.39.a - 6.39.d shows the filleting zone definition with 1 to 4 ranges neighbour nodes and figures 6.39.e - 6.39.h shows the result of the deformation on the corresponding models. It is clear that the number of ranges used in the filleting zone represents implicitly the filleting radius.

6.5.4 Surface filleting zone deformation

The deformation tool adopted to get the smooth shape by repositioning the mesh nodes is presented in the section 4.5 (p.129). In case of a triangle mesh, the local deformation is directly applied on the filleting zone. For a tetrahedral mesh, the deformation is applied in two stages: one on the external hull surface mesh and one in the internal volume mesh. The reason to separate the surface deformation and the volume deformation is that, since the adopted deformation tool is based on a mechanical model, it is better to do not take into account the forces on the internal edges during the external hull surface mesh deformation. Therefore the surface mesh of the tetrahedral mesh is extracted to deform.

During the deformation, all nodes in the filleting zone can move whereas the other nodes are blocked. In case there are groups on the mesh model, the VGBs are computed at preliminary stages. Therefore for preserving the group definition, the nodes in the filleting zone and also on the VBGs will be constrained to stay on VGBs.

The deformation of the filleting zone aims at smoothing the variation of the normal from the filleting zone to the neighbourhood. Therefore the minimization used in the deformation will be the minimization of the sum of the forces applied on the free and blocked nodes (section 4.5.3 on page 136).

6.5.5 Volume mesh relaxation in the filleting zone

For a surface mesh, the filleting stops at the surface mesh deformation as presented in the previous subsection whereas for a volume mesh solely the external skin is deformed for achieving the objective shape. Therefore the internal nodes of the filleting should be moved/relaxed according to the new external nodes position. Therefore, a second deformation step minimizes the sum of the external forces applied on all the free internal nodes in the filleting area.

A very simple example of two-steps deformation is shown in figure 6.40. Suppose the figure concerns a section view of a part of tetrahedral mesh that is defined as a filleting area. In figure 6.40.a there are some internal nodes and external nodes. One can easily imagine that the sharp edges go through the "sharp nodes". The two steps of deformation are performed sequentially and are

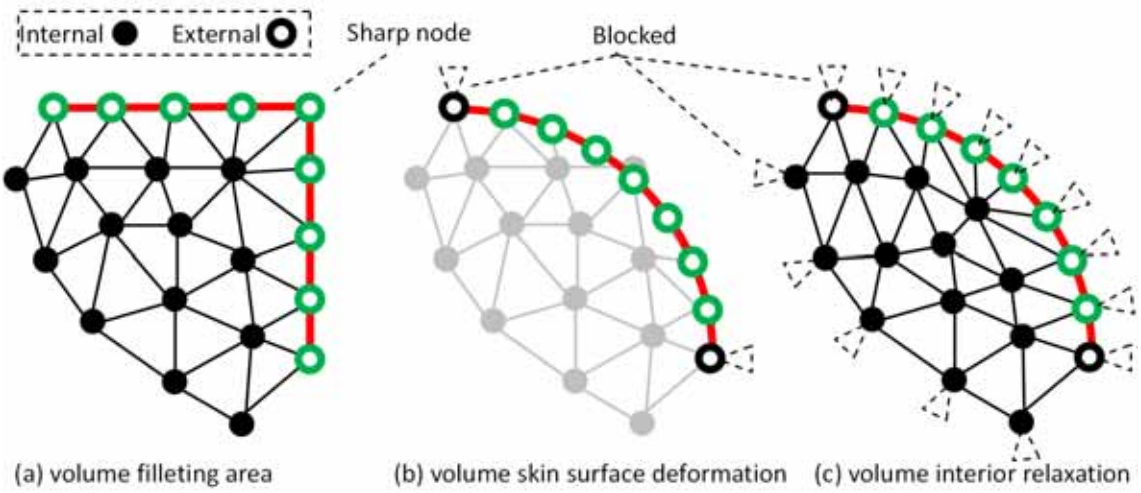


Figure 6.40: Two-steps deformation for tetrahedral mesh filleting

shown in the pictures b) and c). In figure 6.40.b only the external nodes on the model skin are concerned for smooth shape deformation and the two extremities are blocked nodes. The deformation minimizes the sum of the external forces both on the free nodes (green) and blocked nodes (black). All the external nodes except for the two extremities have been moved to render a rounded shape. The internal nodes are not concerned in this deformation and are shown by the gray color. The second deformation step is shown in figure 6.40.c and consists in moving all internal nodes except for the ones on the filleting zone boundary. The deformation in this step minimizes the sum of the external forces applied on the internal free nodes so that these nodes move for relaxation. No constraints are applied here, and the optimisation problem consists in a minimization.

To preserve the group definition, the free nodes should be constrained to stay on the VGBs. Due to time constraints this possibility has not yet been prototyped.

Figure 6.41 shows the CAD-less filleting experimentation performed on the partially rounded parallelepiped that is already presented in figure 6.41. Figure 6.41.a shows the filleting area definition with 6 ranges neighbourhood around sharp edges. Figure 6.41.b shows the result of the filleting. On this example, the filleted zone under CAD system and the filleted zone under prototyped CAD-less operator are clearly shown together. Figures 6.41.c and 6.41.f show the

extracted filleting area before and after the filleting operation. The surface and volume mesh of the filleting area are also shown and used to do the two-stages deformation. The two deformation stages are also shown in figures 6.41.i, 6.41.j and 6.41.k. It consists in an internal view of the model during the two stages. The intermediate deformation on the model skin moves the boundary nodes onto a smooth shape whereas some internal mesh elements (tetrahedra) are outside the model. Therefore the second deformation stage allows relaxing the internal mesh elements by blocking the boundary nodes in order to keep the rounded shape. Accuracy values are given in table 6.11.

6.5.6 Additional experimentations with the mesh filleting operator

The first presented CAD-less filleting experimentation is applied on a tetrahedral mesh on which a discontinuity of the surface appears. In this case, the discontinuity vanishes in the two planar faces located at the extremities. The objective is to smooth this discontinuous area. Figure 6.42.a shows the model on which the sharp edges representing the discontinuity are identified. Figure 6.42.b shows that the filleting area is defined 3 neighbour ranges around the sharp edges. The filleting deformation is shown in figure 6.42.c. Here again, the deformation process consists of two stages. The first stage deforms the boundary surface of the filleting area in order to get a round shape. Figure 6.42.d shows the initial model on which the sharp feature is present. Figure 6.42.e shows that the first stage deformation move the boundary surface nodes of the filleting area onto a round shape in order to smooth the shape. The fact that solely the nodes on the boundary surface of the tetrahedra are moved produces stretched tetrahedra (fig.6.42.e). Therefore the second deformation relaxes the tetrahedra within the filleting area (fig.6.42.f).

The second CAD-less filleting experimentation concerns a hook tetrahedral mesh model (fig.6.43.a). The sharp feature to fillet is between the hook body and the handle part (fig.6.43.b). The filleting area is defined with 2 ranges of neighbours surrounding the sharp edges (fig.6.43.c). The filleting deformation result is shown in figure 6.43.d. From internal view the two stages deformation is

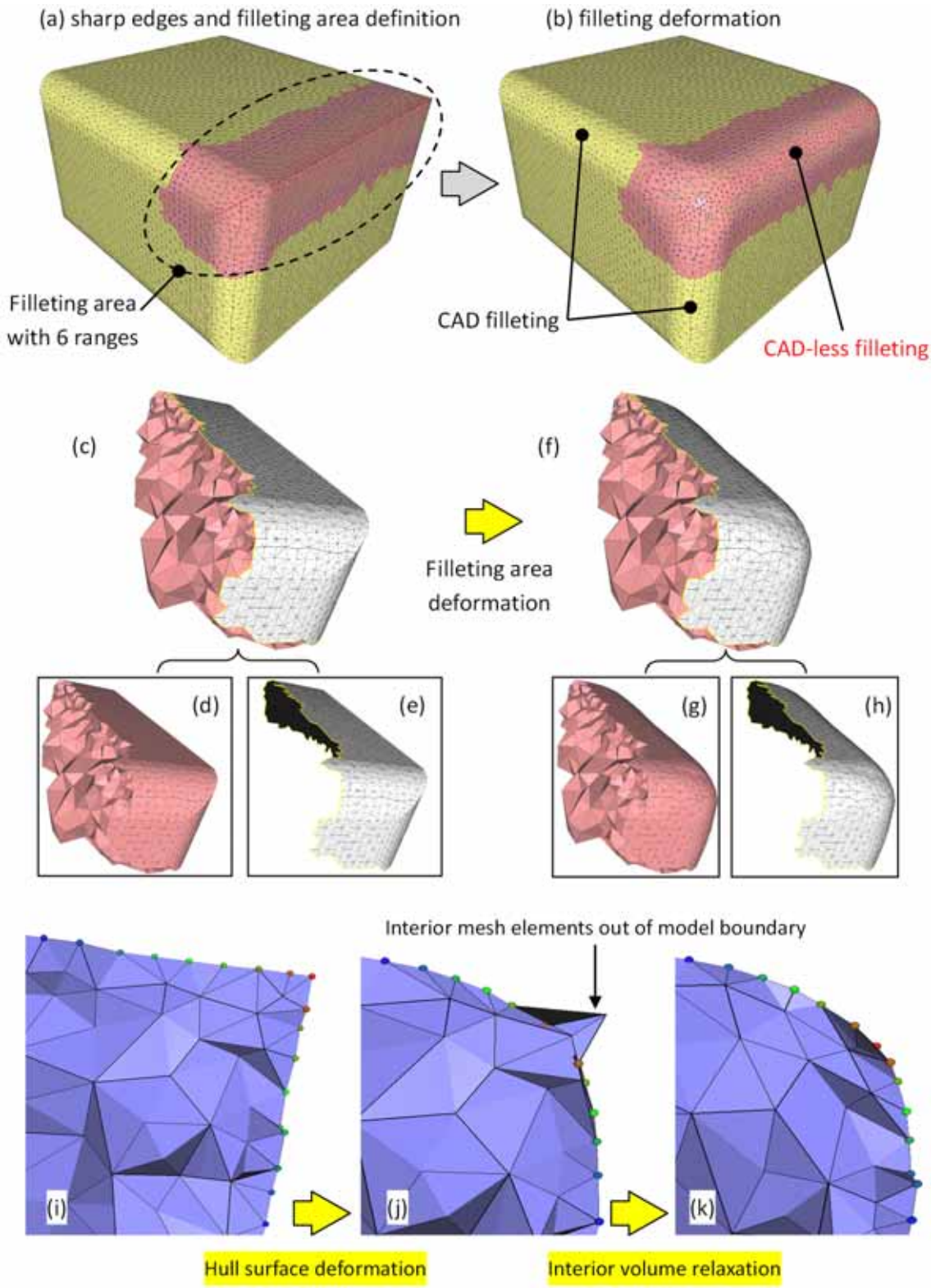


Figure 6.41: Example of tetrahedral mesh filleting

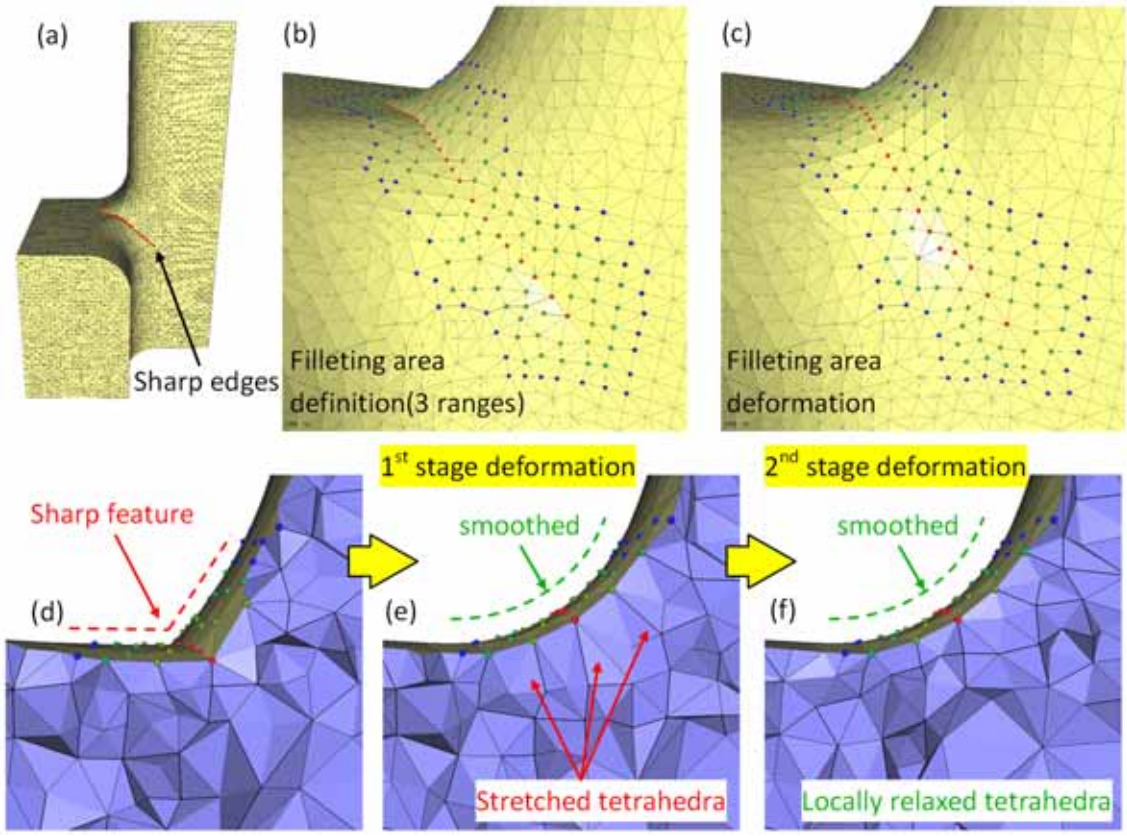


Figure 6.42: CAD-less filleting on a tetrahedral mesh

shown in figures 6.43.e - 6.43.g. The first deformation stage smoothes the model boundary surface (fig.6.43.f) and the second deformation stage (fig.6.43.g) relaxes the interior mesh elements that were stretched during the first stage.

The third tetrahedral model used to experiment the prototyped mesh filleting operator is a half piston. Figure 6.44.a presents the entire piston for which the half-part is used. The sharp edges are shown in Figure 6.44.b and c. They form a multi-branches configuration. The deformation result is shown in figure 6.44.d.

For various experimentations, the nodes used in the two-stages deformation process are quantified in the table 6.10. The surface filleting nodes are the ones repositioned during the first stage deformation whereas the volume relaxation nodes concerns the nodes repositioned during the second stage deformation.

A summary of quality evolution for each experimented models is shown in table 6.11. Again, it shows that some post-processing modifications are required

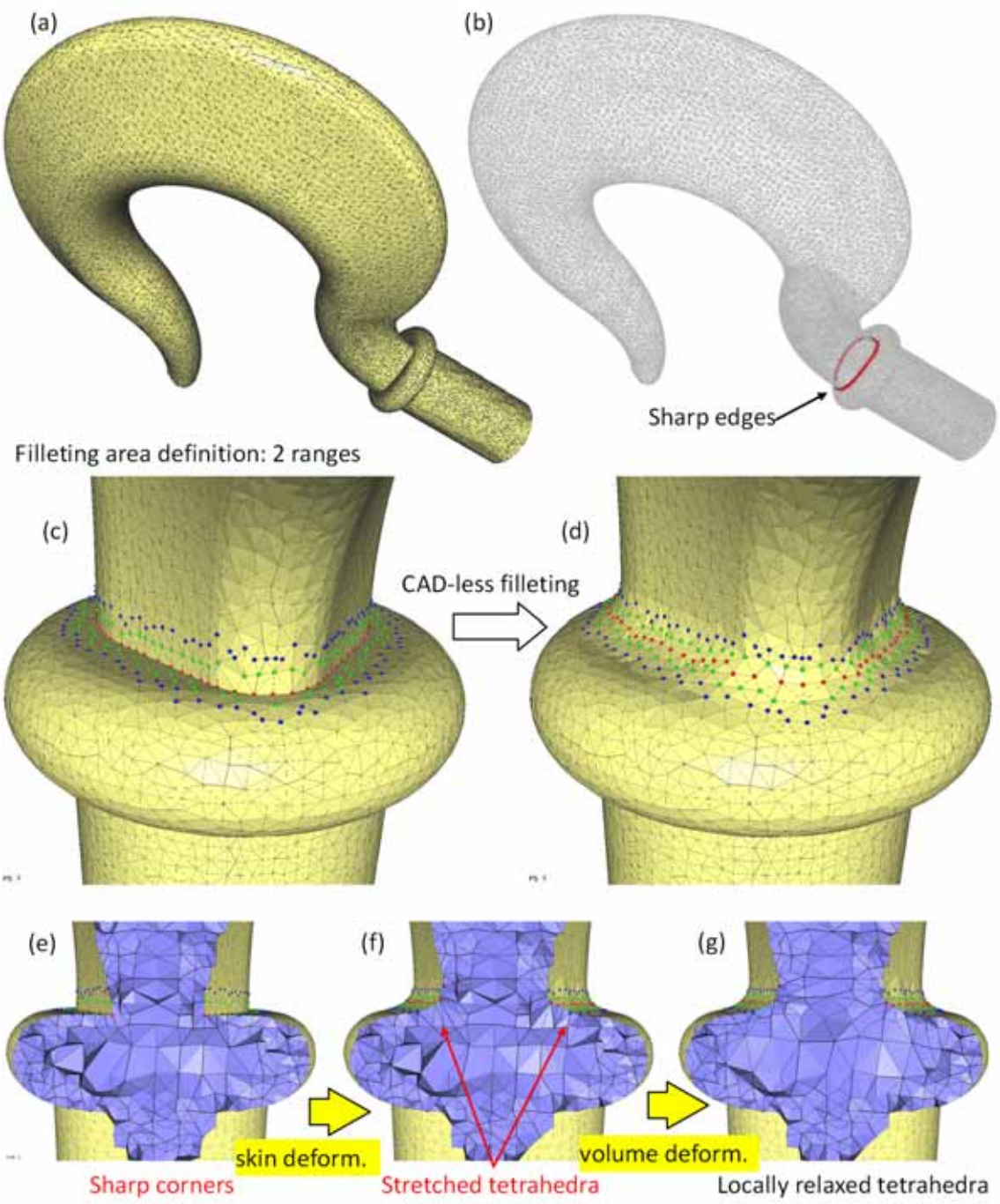


Figure 6.43: Tetrahedral mesh filleting of a hook

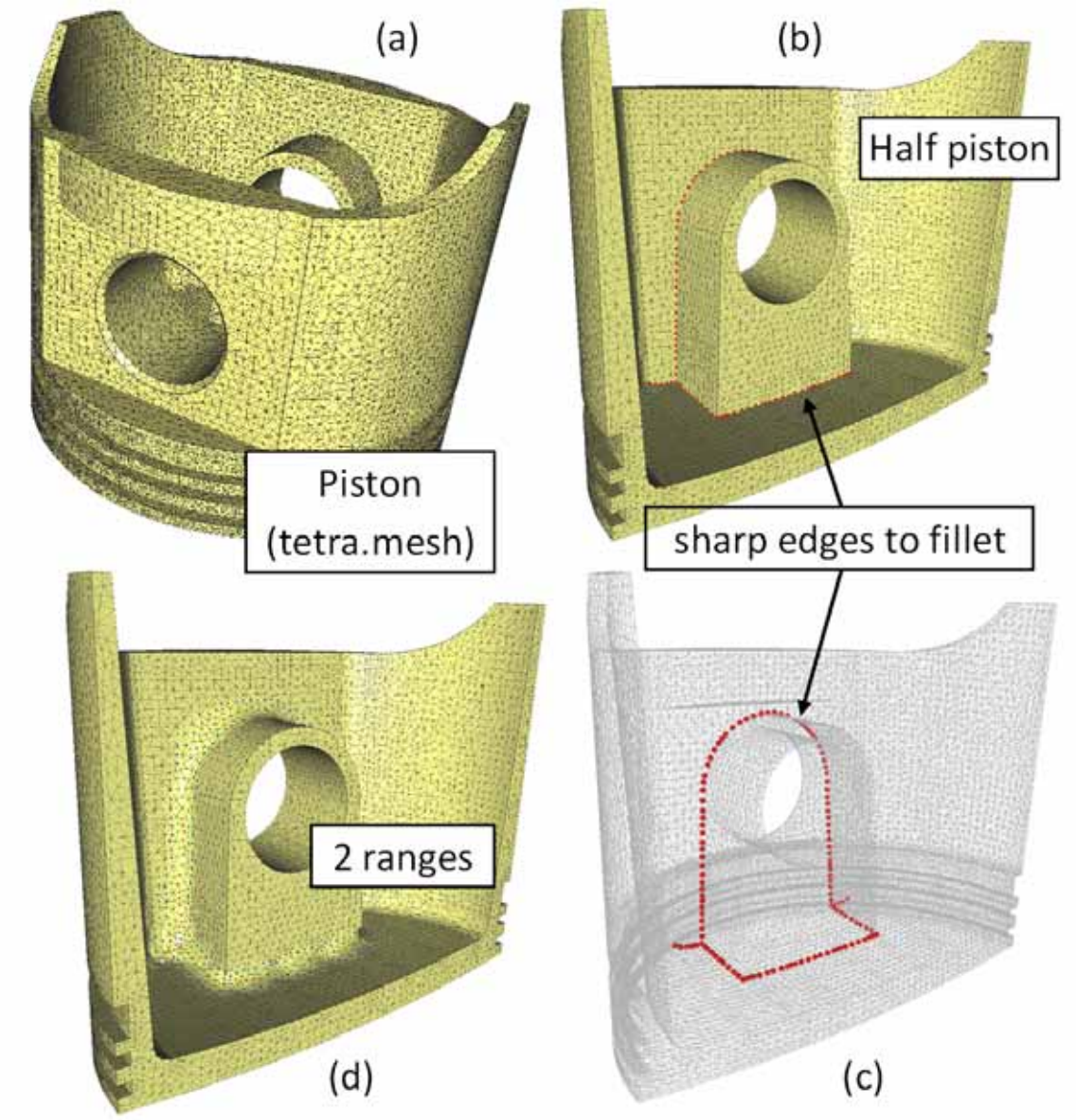


Figure 6.44: Example of tetrahedral mesh filleting on a half-piston

Nodes \ Models	Box partially filleted fig.6.41	Discontinuity fig.6.42	Hook fig.6.43	Piston fig.6.44
Total	27579	28921	41486	40581
Sharp edges	60	22	68	123
Surface filleting nodes	866	245	386	677
Volume relaxation nodes	3833	1864	3265	7460

Table 6.10: Various deformation nodes for filleted models

Aspect ratio Q		Box partially filleted fig.6.41	Discontinuity fig.6.42	Hook fig.6.43	Piston fig.6.44
Initial	Min	0.074	0.410	0.326	0.332
	Max	0.985	0.996	0.976	0.981
	Mean	0,656	0,646	0.647	0.655
~~Cracked~~	Min	0.001	0.09	0.02	0.09
	Max	0.985	0.996	0.974	0.981
	Mean	0,652	0,646	0.646	0.656

Filleted

Table 6.11: Comparison of the aspect ratio Q for the various filleted models

and have not been considered in this manuscript. Further investigations have also proved that while suppressing some elements in the deformation area prior to the hull deformation, the obtained elements are less degenerated.

6.6 Conclusion

In this chapter, several prototyped instances of the generic CAD-less mesh modelling operator have been presented. The prototyped CAD-less operators could be further optimised to improve the mesh quality or the efficiency in terms of the time used to get the results. Nevertheless, the examples clearly show the interest of developing CAD-less mesh modelling operators to avoid going back to the CAD design phase during the optimisation phases. The prototyped operators show also how to instance each operator under the framework of the CAD-less mesh modelling operator that is presented in chapter 3 (p.69).

Since the generic CAD-less mesh modelling operator is modular, i.e. it consists of different blocks of basic tools and methods for processing the geometric models and associated semantics, it is very easy to change the design of each prototyped

CAD-less operators while modifying directly the blocks. In this thesis, different adopted blocks to prototype the mesh modification operators are presented in the previous chapters. Stome tools and methods have been chosen because they were already developed in the LSIS laboratory. Anyhow, it may be very easy to get other blocks from open source websites. According to the preferences of researchers and capabilities of the existing algorithms, these basic tools and methods could be easily changed or optimised to prototype new CAD-less mesh modelling operators taking into account specific needs of a given application.

Synthesis, conclusions and perspectives

In conclusion, this PhD thesis has introduced an industrial context in which different limits and shortcomings during the engineering design and maintenance studies are identified. It has been showed that the classical FEA loop necessitating different steps from CAD modelling until FE simulation is not always appropriate for product maintenance context in which fast assessment of behaviour of the production machinery interacting with real environment and evaluation of different design solutions improving its behaviour are required. In order to avoid useless and time-consuming activities' repetitions, alternative desing solutions should be obtained directly on the FEA semantics enriched meshes. With existing mesh manipulation capabilities several criteria in terms of geometry and semantics maintenance and propagation are not satisfied. To overcome these limits this thesis proposes a new CAD-less framework. The proposed framework is flexible since it is modular and composed by different tools for geometric and semantics treatment of the FE mesh model. For each module the involved tools and methods are presented in details. These tools are then used to prototype four CAD-less mesh modelling operators: mesh merging, cracking, drilling and filleting taking into account quality criteria particularly required for FE meshes. The proposed operators demonstrate concretely the CAD-less framework feasibility and show the methodology for developing new operators under this framework.

The proposed CAD-less framework opens new research perspectives in the sub-domain of semantics-based geometric modelling that could be faced in the future.

Limits and shortcomings of actual engineering

A general industrial engineering problem has been drawn out in this thesis. The classical product design/optimisation numerical loop by using FEA is discussed as well as its limits are studied. The main limit is that for assessing each product design solution it is necessary to start from a CAD model which leads to complete meshing and fully insertion of FEA semantics. The meshing process needs communicating with the CAD model in order to make it meshable. The mesh should have different size in different area according to aim of a given study. Some special non-manifold characteristics required for particular FE simulations should be also inserted such as double meshentities. The FEA semantics insertion can be quite difficult, as different mesh groups for supporting these semantics should be firstly created. Some mesh groups are either aggregated based on CAD groups or specific CAD features called partitions or selected semi-manually from the mesh by using different criteria. FEA semantics are then associated with the corresponding mesh groups. Sometime, the FE simulation model based semantically enriched mesh should be tuned before being used to assess the product design/optimisation solution. The FE model tuning is long-time iterative process. For physical tuning the numerical model, experimental measurements characterising the real behaviour of the structure to be modelled are required. The tuning process iterates the workflow from CAD modelling and mesh generation until numerical/experimental result correlation. Once the FE model based on semantically enriched mesh is validated, the study could be started for assessing design solutions to improve the behaviour of the structure. In case several design solutions are considered to assess from which a best one is chosen, the assessing process will be multiplied.

These limits of the classical FEA loop are identified through fast maintenance and lifecycle analysis studies of the EDF Group. In the nuclear power production domain the engineers have to deal with the maintenance problems on the production equipment qualified and installed. They have very little time to find out the bestmaintenance/optimisation solution respecting all safety criteria in order to minimise the production stop delay. With the classical FEA loop for assessing the various faced solutions the study time becomes too long that has an impact

financially very significant for power production.

Challenge of faster, easier and accurate modelling approach

To overcome these limits a new framework for **CAD-less** FEA approach has been introduced for fast maintenance study of machinery's physical behaviour and prototyping of design solutions during the product optimisation phase. With the proposed approach, the geometric modifications required to achieve the optimal solution can be directly performed on the FE enriched mesh models. This avoids the need to return to the CAD model and then to re-perform all the time consuming steps necessary to obtain the complete FE model. This capability can be also useful when the CAD model is not available and a mesh could be obtained easily, e.g. through object scanning. Therefore, different alternative design solutions could be realised on the available and mechanically valid FE mesh models. To fully exploit such a capability it is important that not only the shape aspects are considered but also the physical simulation semantics present in the FE mesh are correctly managed. Depending on the modification on the shape and on the specific simulation semantics' meaning, the attached semantic data may need different treatment.

Several **criteria** are defined on CAD-less mesh modelling operators in order to satisfy various industrial needs. These criteria are considered from two points of view: geometric and semantic. From geometric point of view, the modification made on the geometric model should be as much local as possible in order to guarantee the validity of the updated (or tuned) FE model end so the FE mesh used; thus the initial shape has to be preserved as much as possible; the quality of produced mesh elements should be taken care of; the self-intersection on the level of mesh should be avoided; the shape of the modification tool should be respected. With regard to the semantic aspect, the definition of different mesh groups on the geometry should be preserved, and the semantics associated with the simulation model should be updated/transformed/propagated according to

the new geometric configuration.

By analysing the current capabilities and research results on mesh manipulation, few works could respect to all of these criteria. Therefore, a new CAD-less mesh modelling framework is proposed in the thesis. The framework is designed as modular so that the development of new operators simply requires the substitution and/or addition of new components or constraints. In this PhD work, different methods and tools are presented and used to prototype different framework instances. The basic methods and tools are subdivided into two above aspects: geometric and semantic one, and are exploiting a three layer information structure: geometry, groups and simulation semantics. The group layer is meant to provide the link between the geometric and the simulation semantic data by accurately arrainging the geometric information.

In geometric aspect, the presented methods and tools are shape recognition, sharp edges detection, topological modifications and deformation. The shape recognition allows providing constraints from initial model shape during the mesh modification. The sharp edges detection allows finding the edges that are used in the filleting operation. The sharp edges detection allows also to impose constraints from characteristic curve on the initial mesh model during the mesh modification. The topological modification allows generating, removing, subdividing and duplicating mesh elements that would be useful in different CAD-less mesh modelling framework instances. The mesh deformation consists in repositioning the nodes in order to realise a certain shape under constraints by preserving a given shape.

From semantic point of view, the proposed tools strongly exploit the group information layer and thus involve group boundary computation, overlapping groups decomposition, and semantics preservation as well as transformation during the mesh modification. The group boundary computation allows providing constraints from the group shape on the initial mesh during the mesh modification process. The overlapping groups' decomposition allows transforming a constrained meshing problem into elementary hole filling problems and it allows also assisting the semantics reassignment during the re-meshing process. The semantics preservation consists in a method to impose the group shape constraints into geometric mesh modifications such as re-meshing, deformation, etc. The

semantic transformation concerns a set of rules about semantics location (reduction/propagation) and parameter value/direction changing during and after geometrical mesh modification.

In order to concretely demonstrate the proposed CAD-less mesh modelling framework, four operator instances have been prototyped. These operators are using the presented basic tools acting on both the geometric and the semantic aspects. Theprototyped operators are: meshes merging, crack/contact sone inserting, cylinder hole drilling and sharp edge filleting, and its have been proposed here to cover the primary needs in terms of direct modification of semantically enriched 2D/3D meshes. These operators use a deformation engine applying two types of constraints: those relative to the shape of the tool (e.g. cylinder for holes and plane for cracks) and of the object itself and those relative to the group boundaries associated to physical semantics that have to be preserved. Since our approach is aimed at complying with the maintenance study context in which it is important to deal with FE meshes that correctly reflect the real behaviour of the studied equipment physical counterpart already validated, we reduce as much as possible the existing mesh elements on which the modifications are applied. Thus, the mesh is modified only in a restricted bandwidth so that the quality of the elements (e.g. their aspect ratio) remains satisfactory with respect to the FEA requirements. Additionally, the proposed operators reposition nodes without adding new ones. Only in special cases, when the operated mesh does not have enough nodes in the modification area, a mesh pre-refinement step is necessary.

Perspectives

Future works can be identified in geometric, group and semantics levels and includes either scientific research or engineering development. The different basic tools and methods used in the CAD-less mesh modelling framework could be improved.

In **geometric** level, the shape recognition technique could be assisted by the sharp edge detection technique. That's to say that sharp edges detection could

help to delimitate sub-patchs on it is applied the shape as well as its parameter identification process. The deformation process could be combined with the topological modification process in order to produce better quality mesh elements. Because for producing a certain shape on a certain surface, solely repositioning the nodes are not sufficient to obtain good quality elements. For example, when the two edges forming a 60° are deformed into be forming 180° if no new edge is inserted between these two edges the concerning triangles will not be equilateral. Reversely if two edges angle is reduced the topological modification should intervene by removing edges.

In **group** level, the VGB (virtual group boundary) notion should be extended. For instance, the bounding elements are computed based on the domain/spatial elements of dimension equal to the one of the mesh. Therefore, for example, a group of triangles in a 3D mesh are considered as isolated elements if they do not enclose any tetrahedron. The notion of bounding elements should be extended into multi-dimensions in order to avoid as much as possible the isolated elements. The motivation of reduce the number of isolated elements is to give more mesh repositioning freedom which could help to improve the mesh quality during the geometric modification.

In **semantics** level, the transfer should be furthermore developed. More rules of semantics propagation in correspondence to the geometric modification and semantics information should be defined. Semantics value changing in specific modification should be developed. Additional case studies should be analysed to strengthen the validity of the semantics transfer rules.

The prototyped **CAD-less mesh modelling operators** could be improved and extended. Other new operators could be developed. For instance, the prototyped operators produce still some bad quality mesh elements. Possible improvements could be at first the one mention before: the combination of deformation process with topological modification. Additionally, post-processing could be considered that could, for example, iterate the steps topological modification, relaxation and quality control. The developed operators could be also extended for example by considering operating tool of different shapes, e.g. the drilling operator could use other shape than cylinder for cutting tool. In addition, other CAD-less new operators should be developed under the framework according to

different industrial needs.

We can state that the results of the research here described give a new vision about the needs and possibilities to develop fast shape modelling tools under different industrial constraints. It enforces the importance of defining modelling operators that are not only dealing with the geometric aspects of the objects but fully use the attached application specific semantics in order to provide tools that really improve the product lifecycle development.

References

[1] V. Adams and A. Askenazi. *Building Better Products with Finite Element Analysis*. OnWord Press, 1. edition, 1999. 28

[2] Max K. Agoston. *Computer Graphics and Geometric Modelling: Mathematics*. Springer; 1st Edition. edition, 2005. 6, 9

[3] Aim@Shape. European network of excellence, key action: 2.3.1.7, semantic-based knowledge systems, fp6, http://www.aimatshape.net. 65

[4] V. S. Alagar, T. D. Bui, and K. Periyasamy. Semantic csg trees for finite element analysis. *Comput. Aided Des.*, **22**:194–198, May 1990. 66, 67

[5] C. G. Armstrong, D. J. Monaghan, M. A. Price, H. Ou, and J. Lamont. *Integrating CAE concepts with CAD geometry*, pages 75–104. Civil-Comp press, Edinburgh, UK, UK, 2002. 65

[6] D. N. Arnold, A. Mukherjee, and L. Pouly. Locally adapted tetrahedral meshes using bisection. *SIAM J. Sci. Comput.*, **22**:431–448, February 2000. 26

[7] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.*, **22**:181–193, March 2006. 87

[8] Timothy J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained delaunay triangulation. *Engineering with Computers*, **5**:161–175, 1989. 21

[9] A. Bargier, F. Giannini, R. Lou, and J-P. Pernot. Surface primitive recognition. Technical report, CNR-IMATI 08/09, 2009. 87

[10] Roshdy S. Barsoum. Application of quadratic isoparametric finite elements in linear fracture mechanics. *International Journal of Fracture*, **10**:603–605, 1974. 10.1007/BF00155266. 60

[11] E. Bechet, H. Minnebo, N. Moes, and B. Burgardt. Improved implementation and robustness study of the x-fem for stress analysis around cracks. *International Journal for Numerical Methods in Engineering*, **64**[8]:1033–1056, 2005. ~~cited By (since 1996) 65~~. 60

[12] M. Bern and P. Plassmann. Mesh generation. In *Handbook of Computational Geometry. Elsevier Science*, pages 291–332, 2000. 120

[13] R. Bidarra and W. F. Bronsvoort. Semantic feature modelling. *Computer-Aided Design*, **32**:201–225, 2000. 65

[14] D. Bielser, V.A. Maiwald, and M.H. Gross. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum*, **18**[3]:C37–C38, 1999. ~~cited By (since 1996) 23~~. 61

[15] H. Biermann ~~and~~ D. Kristjansson ~~abd~~ D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 185–194, New York, NY, USA, 2001. ACM. xi, 52, 67

[16] J. Rossignacand P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, 1993. 25

[17] I. C. Braid. The synthesis of solids bounded by many faces. *Commun. ACM*, **18**:209–216, April 1975. 12

[18] D. Bremberg and G. Dhondt. Automatic crack-insertion for arbitrary crack growth. *Engineering Fracture Mechanics*, **75**[3-4]:404–416, 2008. ~~cited By~~ ~~(since 1996) 4~~. xi, 59, 67

AND

[19] C. D. Bruyns, S. Senger, A. Menon, K. Montgomery, S. Wildermuth, and R. Boyle. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The Journal of Visualization and Computer Animation*, **13**[1]:21–42, 2002. 61

[20] M. Campen and L. Kobbelt. Exact and robust (self-)intersections for polygonal meshes. *Computer Graphics Forum*, pages 397–406, 2010. 55

[21] Scott A. Canann, Yong-Cheng Liu, and Anton V. Mobley. Automatic 3d surface meshing to address today's industrial needs. *Finite Elements in Analysis and Design*, **25**:185–198, March 1997. 22

[22] P. Chopra and J. Meyer. Tetfusion: an algorithm for rapid tetrahedral mesh simplification. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 133–140, Washington, DC, USA, 2002. IEEE Computer Society. x, 24

[23] R. Chouadria and P. Veron. Identifying and re-meshing contact interfaces in a polyhedral assembly for digital mock-up. *Eng. with Comput.*, **22**:47–58, August 2006. xi, 58, 186

[24] Philippe G. Ciarlet. *Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. 120

[25] P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral meshes with accurate error evaluation. In *Proceedings of the conference on Visualization '00*, VIS '00, pages 85–92, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press. 23

[26] Code_Aster. Open source software package for civil and structural engineering finite element analysis and numeric simulation in structural mechanics, `http://www.code-aster.org`. 66

[27] L.C.G. Coelho, G. Coelho, M. Gattass, and L. H. De Figueiredo. Intersecting and trimming parametric meshes on finite-element shells. *International Journal for Numerical Methods in Engineering*, **47**:777–800, 2000. xi, 55, 56, 67

[28] J. Cohen, M. Olano, and D. Manochah. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 115–122, New York, NY, USA, 1998. ACM. 24

[29] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Trans. Graph.*, **23**:905–914, August 2004. 25

[30] M. Dakowicz and C. Gold. Interactive tin modification with a cutting tool. In *Proceedings 4th ISPRS Workshop on Dynamic and Multi-dimensional GIS*, pages 5–9. Pontypridd, Wales, UK, 2005. xi, 62

[31] L. De Luca, P. Veron, and M. Florenzano. Semantic-based modeling and representation of architectural buildings. In *Semantic Virtual Environments*, Villars, Suisse, 2005. 66

[32] L. De Luca, P. Veron, and M. Florenzano. A generic formalism for the semantic modeling and representation of architectural elements. *Vis. Comput.*, **23**:181–205, February 2007. 66, 67

[33] T. De Martino, B. Falcidieno, and F. Giannini. A reference model for product information sharing in concurrent engineering environment. In *Virtual Prototyping, Virtual Environments and the product design process : IFIP WG 5.10 workshops on virtual environments and their applications and virtual prototyping*, pages 146 – 155, 1995. 65

[34] Boris N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, **7**[6]:793–800, 1934. 17, 20

[35] FOCUSK3D. Foster the comprehension and use of knowledge intensive 3d media, http://focusk3d.eu. 65

[36] J.D. Foley, A. Van Dam, S.K. Feiner, and J.F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2. edition, 1990. 14

[37] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from manifold tetrahedralisation: Application to real-time surgical simulation. *Medical Image Analysis*, **9**[2]:113–122, 2005. ~~cited By (since 1996) 12.~~ 61

[38] P. L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering*, **92**:269–288, November 1991. 21

[39] J.L. Graysmith and C.T. Shaw. A fast triangle to triangle intersection test for collision detection: Research articles. *Engineering Computations: Int J for Computer-Aided Engineering*, **14**:702 – 717, 1997. 55

[40] L Guillaume, D. Florent, and B. Atilla. Curvature tensor based triangle mesh segmentation with boundary rectification. In *Proceedings of the Computer Graphics International*, pages 10–17, Washington, DC, USA, 2004. IEEE Computer Society. 109

[41] O Hamri and J-C Leon. Interoperality between cad and simulation models for cooperative design. In *CIRP design seminar*, pages 146 – 155, May 12-14th 2003. 65

[42] O. Hamri, J.-C. Leon, F. Giannini, and B. Falcidieno. Using cad models and their semantics to prepare f.e. simulations. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - DETC2005*, **3 A**, pages 129–138, 2005. ~~cited By (since 1996) 1~~. 65

[43] Lasse Holmström. Piecewise quadric blending of implicity defined surfaces. *Comput. Aided Geom. Des.*, **4**:171–189, November 1987. 63

[44] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface

reconstruction. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 295–302, New York, NY, USA, 1994. ACM. xii, 109, 110, 111

[45] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 19–26, New York, NY, USA, 1993. ACM. x, 23, 24

[46] S-M. Hu, C-F. Li, and H. Zhang. Actual morphing: a physics-based approach to blending. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, SM '04, pages 309–314, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. 64

[47] A. Hubeli, K. Meyer, and Gross M. Mesh edge detection. Technical report, ETH Zurich, CS Technical Report #351, December 4, 2000. xii, 110, 111, 112, 113

[48] T. Igarashi and J. F. Hughes. Smooth meshes for sketch-based freeform modeling. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, I3D'03, pages 139–142, New York, NY, USA, 2003. ACM. 54

[49] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *SIGGRAPH'99*, pages 409–416, 1999. 54

[50] M.S Shephard J-F Remacle. An algorithm oriented mesh database. *International Journal for Numerical Methods in Engineering*, **58**:349–374, 2003. 16

[51] Z. Ji, L. Liu, Z. Chen, and G. Wang. Easy mesh cutting. *Computer Graphic Forum (Proceedings of Eurographics)*, **25**[3]:283–291, 2006. 61, 202

[52] X. Jin, J. Lin, C.C.L. Wang, J. Feng, and H. Sun. Mesh fusion using functional blending on topologically incompatible sections. *Visual Computer*, **22**[4]:266–275, 2006. 57

[53] Wonhyung Jung, Hayong Shin, and Byoung K. Choi. Self-intersection removal in triangular mesh offsetting. In *Computer-Aided Design and Applications*, **1**, pages 477–484, 2004. xi, 56

[54] S.J. Kim, D.Y. Lee, and M.Y. Yang. Offset triangular mesh using the multiple normal vectors of a vertex. In *Proceedings of Computer-Aided Design and Applications*, 2004. xi, 64, 65, 67

[55] Premysl Krsek. Complex human tissues fem models prepared by boolean operations. In *Biomechanics of man 2002*, pages 24–26. Faculty of Physical Education and Sport Charles University in Prague, 2002. 52, 67

[56] K. Kundu and M. Olano. Tissue resection using delayed updates in a tetrahedral mesh. *Studies in health technology and informatics*, **125**:241–243, 2007. ~~cited By (since 1996) 0.~~ 61

[57] T. S. Lau, S. H. Lo, and C. K. Lee. Generation of quadrilateral mesh over analytical curved surfaces. *Finite Elements in Analysis and Design*, **27**:251–272, November 1997. 22

[58] C.L. Lawson. Software for c1 surface interpolation. *Rice, J.R. (Ed.), Mathematical Software III, Academic Press*, pages 161–194, 1977. 21

[59] D. Lesage, J.-C. Leon, and P. Veron. Discrete curvature approximations and segmentation of polyhedral surfaces. *International Journal of Shape Modeling*, **11**[2]:217–252, 2005. ~~cited By (since 1996) 1~~. xii, 113, 115, 116

[60] Peter Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 200–205, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 26, 121, 123, 195, 196

[61] P. Lindstrom and G. Turk. Image-driven simplification. *ACM Trans. Graph.*, **19**:204–241, July 2000. 25

[62] W.M. Lira, L.C.G. Coelho, and L.F. Martha. Multiple intersections of finite-element surface meshes. In *11th International Meshing Roundtable*, pages 355–366, 2002. 55, 67

[63] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *Math. Comput.*, **65**:1183–1200, July 1996. 26, 27

[64] Y. Liu, C. Xu, Z. Pan, and Y. Pan. Semantic modeling for ancient architecture of digital heritage. *Computers & Graphics*, **30**[5]:800 – 814, 2006. 66

[65] Y. Liu, C. Xu, Q. Zhang, and Y. Pan. Ontology based semantic modeling for chinese ancient architectures. In *Proceedings of the National Conference on Artificial Intelligence*, **2**, pages 1808–1813, 2006. ~~cited By (since 1996) 1.~~ 66, 67

[66] Y-S. Liu, H. Zhang, J-H. Yong, P-Q. Yu, and J-G. Sun. Mesh blending. *The Visual Computer*, pages 915–927, 2005. 64

[67] S.H. Lo. Volume discretization into tetrahedra–i. verification and orientation of boundary surfaces. *Computers & Structures*, **39**[5]:493–500, 1991. 20

[68] S.H. Lo. Volume discretization into tetrahedra–ii. 3d triangulation by advancing front approach. *Computers & Structures*, **39**[5]:501–511, 1991. 20

[69] R. Lohner, P. Parikh, and C. Gumbert. Interactive generation of unstructured grid for three dimensional problems. *Numerical Grid Generation in Computational Fluid Mechanics*, pages 687–697, 1988. 20

[70] Rainald Lohner. Progress in grid generation via the advancing front technique. *Engineering with Computers*, **12**:186–210, 1996. 10.1007/BF01198734. 17, 20

[71] Charles Teorell Loop. *Smooth Subdivision Surfaces Based on Triangles*. Department of mathematics, University of Utah, Aug. 1987. 18

[72] R. Lou, F. Giannini, J-P. Pernot, A. Mikchevitch, Falcidieno B., P. Veron, and R. MARC. Direct modification of ~~fe~~ FE meshes preserving group information. In *Tools and Methods for Competitive Engineering*, TMCE '10, ~~00~~ vol.1, pp.303-315, Italy, April 2010. 202

[73] R. Lou, F. Giannini, J-P. Pernot, A. Mikchevitch, Falcidieno B., P. Veron, and R. MARC. Semantic-preserving mesh direct drilling. In *Proceedings of the 2010 Shape Modeling International Conference*, SMI '10, pages 68–77, Washington, DC, USA, 2010. IEEE Computer Society. 130, 221

[74] R. Lou, F. Giannini, J-P. Pernot, A. Mikchevitch, Falcidieno B., P. Veron, and R. MARC. Direct modification of semantically-enriched finite element meshes. *the International Journal of Shape Modeling*, ~~j~~June 2011. Accepted for publish. 202, 221

[75] R. Lou, J-P. Pernot, A. Mikchevitch, and P. Veron. Direct merging of triangle meshes preserving simulation semantics for fast modification of numerical models. In *Tools and Methods for Competitive Engineering (TMCE'08)*, **1**, pages 119–131, 2008. ISBN 978–90-5155–044-3. 181

[76] R. Lou, J-P. Pernot, A. Mikchevitch, and P. Veron. Merging enriched finite element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance. *Computer-Aided Design*, **42**[8]:670 – 681, 2010. Application-driven Shape Development. 121, 129, 181

[77] A. Martinet, E. Galin, B. Desbenoit, and S. Akkouche. Procedural modeling of cracks and fractures. In *Shape Modeling International SMI 2004*, pages 346–349, 2004. 60

[78] A. Mikchevitch, S. Geniaut, and I. Nistor. Towards fast numerical studies for maintenance and lifecycle problem analysis: New simulation methods on example of an industrial study case. In *American Society of Mechanical Engineers, Pressure Vessels and Piping Division (Publication) PVP*, **3**, pages 253–262, 2010. cited By (since 1996) 0. 60

[79] J. MITANI. A simple-to-implementation method for cutting a mesh model by a hand-drawn stroke. In *Proceedings 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 35–41. Trinity College Dublin, 2005. xi, 62, 63

[80] N. Moes, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, **46**[1]:131–150, 1999. 60

[81] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1142–1147, New York, NY, USA, 2005. ACM. 63

[82] H-W. Nienhuys and A. F. van der Stappen. A surgery simulation supporting cuts and finite element deformation. In Wiro Niessen and Max Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001*, **2208** of *Lecture Notes in Computer Science*, pages 145–152. Springer Berlin / Heidelberg, 2001. xi, 60, 67

[83] M. Panchetti, J-P. Pernot, and P. Veron. Polyhedral simplification preserving character lines extracted from images. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications*, pages 81–90, Washington, DC, USA, 2007. IEEE Computer Society. 25

[84] Sang C. Park. Polygonal extrusion. *The Visual Computer*, **19**:38–49, 2003. 10.1007/s00371-002-0170-2. 57

[85] Sang C. Park. Triangular mesh intersection. *The Visual Computer*, **20**:448–456, 2004. 10.1007/s00371-004-0251-5. 56

[86] D. Pavic, M. Campen, and L. Kobbelt. Hybrid booleans. *Computer Graphics Forum*, **29**[1]:75–87, 2010. xi, 53, 54, 67

[87] S. Peng, R.Lou, and J-P.Pernot. Tetrahedral mesh filleting. Bachelor thesis, Arts & Metiers ParisTech and Beijing University of Aeronautics and Astronautics, June 2010. 109

[88] J-P. PERNOT. *Fully Free Form Deformation Features for Aesthetic and Engineering Designs*. PhD thesis, PhD dissertation in cotutelle entre l'INP Grenoble en France et Universita di Genova en Italie, october 2004. 130, 131, 136, 137

[89] J-P Pernot, G. Moraru, and P. Veron. Filling holes in meshes using a mechanical model to simulate the curvature variation minimization. *Comput. Graph.*, **30**:892–902, December 2006. xiii, 121, 130, 141

[90] A. Plaza and G. F. Carey. About local refinement of tetrahedral grids based on bisection. In *In 5th International Meshing Roundtable*, pages 123–136, 1996. 26, 27

[91] A. Plaza and M-C. Rivara. Mesh refinement based on the 8-tetrahedra longest-edge partition. In *Partition, Proceedings, 12th International Meshing Roundtable, Sandia National Laboratories*, pages 67–78, 2003. x, 26, 27

[92] Ari Requicha. *Geometric Modeling: A First Course*. 1995-1999. 6, 8, 9

[93] M-C. Rivara and G. Iribarren. The 4-triangles longest-side partition of triangles and linear refinement algorithms. *Math. Comput.*, **65**:1485–1502, October 1996. 27

[94] Mara Cecilia Rivara. New mathematical tools and techniques for the refinement and/or improvement of unstructured triangulations. In *Proceedings of 5th International Meshing Roundtable, Sandia National Laboratories*, pages 77–86, 1996. x, 26, 27

[95] D. Rose, K. Bidmon, and T. Ertl. Intuitive and interactive modification of large finite element models. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 361–368, Washington, DC, USA, 2004. IEEE Computer Society. 58

[96] J. Rossignac and A Requicha. A sketch-based interface for detail-preserving mesh editing. In *Proceedings ASME Computers In Mechanical Engineering*, pages 65–73, 1984. 63

[97] Jaroslaw Roman Rossignac. *Blending and offsetting solid models (cad/cam, computational geometry, representations, curves, surfaces, approximation)*. PhD thesis, The University of Rochester, 1985. 63

[98] Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, **18**:548–585, May 1995. 26

[99] SALOME. Open source integration platform for numerical simulation, http://www.salome-platform.org. 66

[100] M. Schollmann, M. Fulland, and H. A. Richard. Development of a new software for adaptive crack growth simulations in 3d structures. *Engineering Fracture Mechanics*, **70**[2]:249–268, 2003. ~~cited By (since 1996) 38.~~ 60

[101] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 65–70, New York, NY, USA, 1992. ACM. 23

[102] M-S. Shephard. and M-K. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, **32**[4]:709–749, 1991. 17, 19

[103] J. R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, SCG '98, pages 86–95, New York, NY, USA, 1998. ACM. 125

[104] A.A. Shostko, R. Lohner, and W.C. Sandberg. Surface triangulation over intersecting geometries. *International Journal for Numerical Methods in Engineering*, **44**[9]:1359–1376, 1999. xi, 57

[105] H. Si. Tetgen: A quality tetrahedral mesh generator and a 3D delaunay triangulator. xiii, 125, 126

[106] H. Si and K. Gartner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In Byron W. Hanks, editor, *Pro-*

ceedings of the 14th International Meshing Roundtable, pages 147–163. Springer Berlin Heidelberg, 2005. 125

[107] J. M. Smith and N. A. Dodgson. A topologically robust algorithm for boolean operations on polyhedral shapes using approximate arithmetic. *Computer Aided Design*, **39**:149–163, February 2007. 54

[108] A. Sourin and A. Pasko. Function representation for sweeping by a moving solid. In *Proceedings of the third ACM symposium on Solid modeling and applications*, SMA '95, pages 383–391, New York, NY, USA, 1995. ACM. 14

[109] O.G. Staadt and M.H. Gross. Progressive tetrahedralizations. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 397–402, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. 23

[110] John Stark. *Product Lifecycle Management: Paradigm for 21st Century Product Realisation*. Springer-Verlag, 1. edition, August 27, 2004. 28

[111] J.R. Tristano, C.D. Zhijan, D.A. Hancq, and W. Kwok. Fully automatic adaptive mesh refinement integrated into the solution process. *ANSYS Inc., Canonsburg PA*, 2003. 25

[112] O. Tropp, A. Tal, and I. Shimshoni. A fast triangle to triangle intersection test for collision detection: Research articles. *Comput. Animat. Virtual Worlds*, **17**:527–535, December 2006. 55

[113] I. J. Trotts, B. Hamann, and K. I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, **5**:224–237, 1999. 23

[114] I.J. Trotts, B. Hamann, K. I. Joy, and D. F. Wiley. Simplification of tetrahedral meshes. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 287–295, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. 23

Volume 43, Issue 7, July 2011, Pages 809-819
The 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling

[115] T.Strouboulis, I.Babuska, and K.Copps. The design and analysis of the generalized finite element method. *Computer methods in applied mechanics and engineering*, **181**[1-3]:43–69, 2000. 60

[116] G. Turini, F. Ganovelli, and C. Montani. Simulating drilling on tetrahedral meshes. In *Proceedings of Eurographics Conference*, pages 127–131. Eurographics, 2006. xi, 61, 67

[117] G. Turkiyyah, W.B. Karam, Z. Ajami, and A. Nasri. Mesh cutting during real-time physical simulation. *Computer-Aided Design*, ~~**In Press, Corrected Proof**, 2010.~~ 61

[118] A. Utsumi, Y. Yagi, and M. Yachida. Estimating surface and spatial structure from wire-frame model using geometrical & heuristical relation. In *Proceedings of IAPR Workshop on Machine Vision Applications, MVA 1992*, pages 25–28, December 7-9 1992. 7

[119] A. Utsumi, Y. Yagi, and M. Yachida. Recognizing surface and spatial structure of environment from wire-frame model obtained by motion stereo model using geometrical & heuristical relation. In *Scandinavian Conference on Image Analysis*, pages 441–447, May 1993. 7

[120] C.C.L. Wang and M.M.F. Yuen. Sketch based mesh extrusion with remeshing techniques. In *Proceedings of the ASME Design Engineering Technical Conference*, **1**, pages 731–738, 2001. ~~cited By (since 1996) 0.~~ 57

[121] Charlie C.L. Wang. Approximate boolean operations on large polyhedral solids with partial mesh reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, **99**~~[RapidPosts]~~, 2010. xi, 52, 53

[122] D-F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, **24**[2]:167–172, February 1981. 21

[123] N. P. Weatherill and O. Hassan. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary

constraints. *International Journal for Numerical Methods in Engineering*, **37**[12]:2005–2039, 1994. 21

[124] M-A. Yerry and Shephard M-S. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, **20**[11]:1965–1990, 1984. 19

[125] D. Zorin and P. Schröder. Subdivision for modeling and animation. Technical report, SIGGRAPH 2000, 2000. Course Notes. x, 17, 18