Extraction of generative processes from B-Rep shapes and application to idealization transformations

Flavien Boussuge, Jean-Claude Léon, Stefanie Hahmann, Lionel Fine

Extraction of generative processes from B-Rep shapes and application to idealization transformations

Flavien Boussuge [a,b,*] Jean-Claude Léon [b] Stéfanie Hahmann [b] Lionel Fine [a]

[a] *EADS Innovation Works, 5, quai Marcel Dassault, 92150 Suresnes, France*

[b] *University of Grenoble / INRIA – JLK laboratory, 655, av. de l'Europe, 38334 Montbonnot, France*

Abstract

A construction tree is a set of shape generation processes commonly produced with CAD modelers during a design process of B-Rep objects. However, a construction tree does not bring all the desired properties in many configurations: dimension modifications, idealization processes,... Generating a non trivial set of generative processes, possibly forming a construction graph, can significantly improve the adequacy of some of these generative processes to meet user's application needs. This paper proposes to extract generative processes from a given B-rep shape as a high-level shape description. To evaluate the usefulness of this description, finite element analyses (FEA) and particularly idealizations are the applications selected to evaluate the adequacy of additive generative processes. Non trivial construction trees containing generic extrusion and revolution primitives behave like well established CSG trees. Advantageously, the proposed approach is primitive-based, which ensures that any generative process of the construction graph does preserve the realizability of the corresponding volume. In the context of FEA, connections between idealized primitives of a construction graph can be efficiently performed using their interfaces. Consequently, generative processes of a construction graph become a high level object structure that can be tailored to idealizations of primitives and robust connections between them.

Key words: B-Rep model, construction graph, generative shape process, idealization

1. Introduction

Shape decomposition is a frequent approach to analyze and structure objects. During product development processes, process planning defining how to sequence machining operations [1,2], FEA evaluating the structural behavior of components [3–5], are task examples that require shape decomposition. The initial generation of the component shape during the design phase can be regarded as the task where the component structure is generated with feature-based B-Rep modelers or hybrid CAD modelers, which also provide Boolean operators. Classically, the object structure is described using a binary tree containing the elementary features or primitives generating the object. This tree structure contains information very specific to each CAD software and, most of the time, is lost when transferring objects across CAD systems and across phases of product development processes. Indeed, the construction tree obtained is very efficient to produce a parameterized representation of the object allowing a user to easily modify some dimensions as well as the shape of the object.

However, the construction tree produced during the design phase may not be well suited for the shape decompositions taking place at other stages of product development processes, e.g. process planning and finite element analyses. As an example, Figure 1 illustrates this configuration with a rather complex component where the entire user's modeling process contains 34 steps, some of them containing multiple contours producing simultaneously several features. Two views defined as top and bottom in Figure 1a show the major details of the component shape. Figure 1b depicts some of the 34 steps involving either extrusion or revolution operations incorporating material addition or removal as complementary effect when shaping the object. As it appears on Figure 1, the generative process can be rather complex and, even if it is available, there is no straightforward use or transformation of this process that could be exploited to idealize this object even though its shape contains stiffeners and thin areas that can be modeled with plate or shell elements rather than volume ones. With re-

* tel. +33 608552947

Email addresses: , Flavien.Boussuge@eads.net (Flavien Boussuge), Jean-Claude.Leon@grenoble-inp.fr (Jean-Claude Léon), Stefanie.Hahmann@inria.fr (Stéfanie Hahmann), Lionel.Fine@eads.net (Lionel Fine).

spect to the idealization objective, it appears mandatory to produce another generative process that could incorporate features or primitives whose shapes are close enough to that of stiffeners and thin wall areas.

Similarly, the current generative process is not well suited if the depth of some stiffener, S_t, needs to be modified (see Figure 1a) because this dimension is obtained through several material removal and addition operations (see steps 18, 20, 23 in Figure 1b).

Using generative processes to model an object shape independently of any CAD modeler is a means to obtain a description that is intrinsic to each object [6]. They stand for a set of modeling actions that can be used to generate different construction histories of this object. All the previous observations illustrate the interest of associating generative processes to not only a history but to several ones to be able to meet various user's needs. To this end, we propose to extract a construction graph from B-Rep CAD models so that the corresponding generative processes are useful for FEA and, particularly, idealization processes. This graph is extracted using a primitive removal operator that simplifies progressively the object's shape. Somehow, the principle is to go 'backward over time'.

The paper is structured with Section 2 that reviews prior contributions. Section 3 sets the context and the hypotheses of the proposed approach. Then, Section 4 describes identification and removal operators of primitives. Section 5 sets up the extraction of the generative process graph used in Section 6 to perform idealizations.

2. Related work

As observed in the introduction, B-Rep decomposition has been extensively studied for feature recognition and suppression. In CAE applications, B-Rep and meshed representations [7–9] have been used to support local simplifications for detail removal. Machining feature recognition has been pioneered by Vandenbrande [9] and has got many contributions to recognize and classify negative features as holes, slots or pockets [10–12]. Woo et al. [1,2] contributed with a volume decomposition approach using a concept of maximal volume and observed that some of them may not be meaningful as machining primitives. In the field of visualization, Lee et al. [13] address a progressive solid model generation. To reduce the complexity of assembly models, [14,15] propose a multi-resolution decomposition of an initial B-rep assembly model. Lockett [16] proposes to recognize specific positive injection molding features. Her method uses an already generated Medial Axis (MA) to find features from idealized models. One recognized difficulty is the ability to obtain MA in a wide range of configurations.

One common obstacle of feature recognition approaches is their difficulty to set feature definitions that can be general enough to process a large range of configurations. This is often mentioned by authors when features are interacting between them because the diversity of interactions can lead to a large range of configurations that can be difficult to identify and structure.

Finding sequences of blend features in an initial shape is relevant to FE preprocessing. Regarding blends removal, Zhu and Menq [17] and Venkataraman [18] detect and classify fillet/round features in order to create a suppression order and remove them from a CAD model. These operations can contribute to the identification of steps of a construction tree.

In FEA, automatic decomposition of mechanical parts into sub-regions create positive feature decompositions. Chong et al. [19] use an edge pairing approach to initiate mid-surface areas with split edges but pairing may not always exist. The face-pairing [20,21] works from nearly parallel faces of CAD models, which produces robust results on a reduced set of configurations. MA Transform (MAT) methods work on mesh models, which is more generic, but produce complex geometry in connection areas. More recently, Robinson and Armstrong [22] use the MA to identify thin regions candidates for idealization.

Still, the idealization processes face difficulties to handle general configurations and connections between idealized sub domains through extension operations. It is effectively the case when mid-surfaces are parallel to each other. Also, mid-surfaces are not always the optimal location of idealized sub domains and external/internal ones are preferred.

The methods of Lu et al. [3] or Liu and Gadh [5] that use edge loops to find convex and sweepable sub-volumes for hex meshing and, more recently, the one proposed by Makem [4] that identifies automatically long, slender regions are other applications of features to FEA. [3] shows that the decomposition criteria must differ from the machining ones.

Shapiro [23] and Buchele [24] address B-Rep to CSG conversion as means to associate a construction tree to a B-Rep model and [24] applies it to reverse engineering configurations. CSG tree representations can be categorized as either halfspace or bounded solid decompositions. In [23,24] B-Rep to halfspace CSG representation is studied and it has been demonstrated that halfspaces solely derived from a volume boundary cannot always be combined into a CSG tree forming a valid solid.

Robinson and al. [25] use preexisting CAD information to identify 2D sketches that are analyzed by MA, which are not always efficient (see section 1 and Figure 1). Yang et al. [26] use a preexisting history tree as basis for model repairing. Li et al. [27,28] introduced a regularity feature tree used to highlight symmetry in the object that differs from CSG and construction trees. Belaziz et al. [29] proposed a morphological analysis of solid models based on features and B-Rep transformations that are able to simplify the shape of an object. It is also a type of B-Rep to CSG conversion. Indeed, the shape modifiers are elementary B-Rep operators that do not convey shape information.

All the approaches generating a CSG type tree structure from a B-Rep bring a higher level of shape analysis. However, the corresponding framework of B-Rep to CSG con-

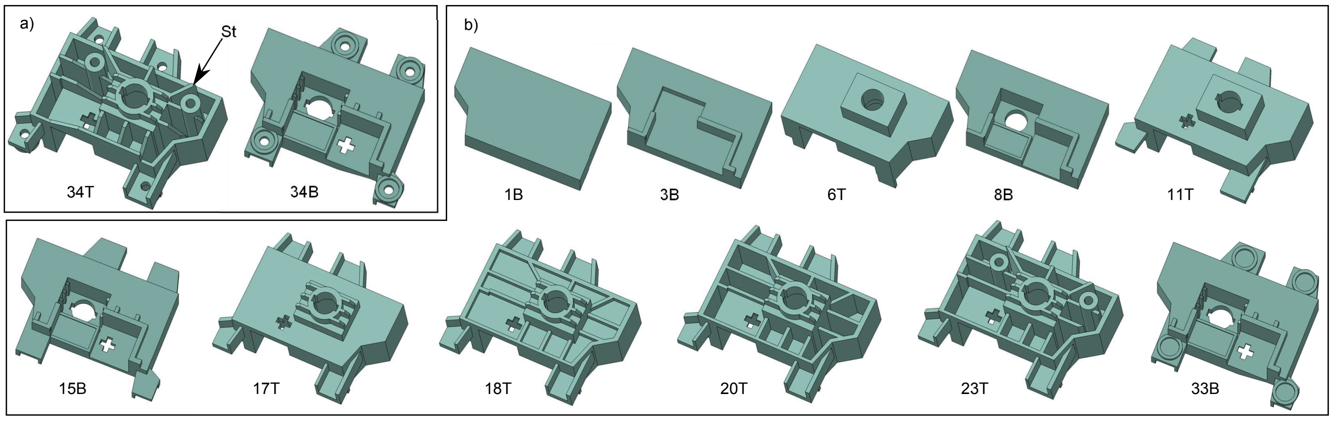


Fig. 1. An example of shape generation process. a) final object obtained after 34 modeling steps and viewed from top (T) and bottom (B). b) some intermediate shapes obtained after the i^{th} modeling step. The letter T or B after the step number indicates whether the object is viewed from top or bottom.

version must be carefully addressed to avoid unresolvable boundary configurations and producing a single tree structure appears to be too restrictive. Achieving good quality connexions between idealized sub domains in a robust manner is still a bottleneck of many approaches processing CAD volumes for FEA.

3. Modeling context and process hypotheses

Construction trees are important because an object submitted to FEA preparation can be subjected to different simplifications at different levels of its construction process. One key issue of these trees is to rely on primitives that are available in common industrial CAD software and to store the sequence of shapes from the simple initial primitive shape to the final object. This behavior is consistent with the objective of using the tree contents for idealization and simplification purposes because the shapes obtained after these operations should get simpler to stay meaningful with regard to these purposes. Detail removal and idealization are tasks that are not entirely formalized. Consequently, the user's know-how has still an important impact on these tasks to adjust the results of idealization and simplification algorithms. Idealization refers to geometric transformations where a subset of a volume can be changed into a surface or a line, i.e. dimensional reductions whereas simplifications are dimension preserving.

The construction tree is a well-known concept for a user. Therefore, the user's know-how can be used to navigate a shape and to select a simpler shape within its history so that it better fits the idealization requirements. As a conclusion, one can state that enabling shape navigation using primitive features similar to that of CAD software is an efficient complement to algorithmic approaches and construction trees or, more generally, construction graphs can support efficiently both.

In our method, the focus is placed on B-Rep mechanical components being designed using volume modelers. Looking at feature-based modeling functions in industrial CAD systems, they all contain extrusion and revolve operations which are combined with addition or removal of volume domains (see Figure 2a). The most generic version of the extrusion, as available in all CAD software, is defined with an extrusion direction orthogonal to a plane containing the primitive contour. Such an extrusion as well as the revolution are defined here as the *reference primitives*. These feature-based B-Rep operations can be seen as equivalent to regularized Boolean operations as available also in common hybrid CAD modelers. Modelers also offer other primitives, to model e.g. draft surfaces, stiffeners, or free-form surfaces from multiple sections. Even though we don't address these primitives here, it is not a limitation of our method. Indeed, draft surfaces, stiffeners and similar features can be modeled with a set of *reference primitives* when extending our method to extrusion operations with material removal and revolutions. Figure 2 illustrates some examples. An extrusion feature where the extrusion direction is not orthogonal to the sketching plane used for its definition. However, the resulting shape can be decomposed into an extrusion orthogonal to a sketching plane and 'cuts', see Figure 2c. The slanted shape of the initial object in Figure 2d is another example of extrusion with material removal.

Another category of form features available from B-Rep CAD modelers are blending radii. Generally, they have no simple equivalence with extrusions and revolutions. Generated from B-Rep edges, they can be classified into two categories:

1- constant radius blends that can produce cylindrical, toroidal or spherical surfaces;
2- constant radius blends attached to curvilinear edges and variable radius blends.

Category 1 blends reflect extrusion and revolution primitives and can be incorporated in their corresponding sketch (see Figure 2a). This family of objects is part of the current approach. Category 2 blends are not yet addressed and are left for future work. Prior work in this field [18,17,30] can be used to derive the object to be analyzed from the initial object, possibly with user's interactions. In summary,

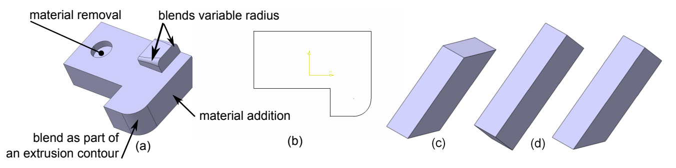


Fig. 2. a) Set of basic volume modeling operators, b) sketch defining an extrusion primitive in (a), c) higher level volume primitive (slanted extrusion), d) reference primitive and its first 'cut' transformation to generate the object in c).

all reference primitives considered here are generated from a sketch step in a plane defining at least one closed contour. The contour is composed of line segments and arcs of circles, (see Figure 2b). This is a consequence of the previous hypothesis reducing the shapes addressed to closed surfaces bounded by planes, cylinders, cones, spheres, tori, and excluding free-form shapes in the definition of the object boundary. This is not really restrictive for a wide range of mechanical components except for blending radii. We therefore restrict the generative processes to extrusion primitives in order to reduce the complexity of the description of the proposed approach in the present paper. Further hypotheses are made in the following sections.

3.1. *Generative process hypotheses and context*

Given a target object M to be analyzed, let us first consider the object independently of the modeling context stated above. M is obtained through a set of primitives combined together by adding or removing material. Combinations of primitives thus create interactions between their bounding surfaces, which, in turn, produce intersection curves that form edges of the B-Rep M. Consequently, edges of M contain traces of generative processes that produced its primitives. Hence, following Leyton's approach [6], these edges can be seen as memory of generation processes where primitives are sequentially combined.

Current CAD modelers are based on strictly sequential processes because the user can hardly generate simultaneous primitives without looking at intermediate results to see how they combine/interact together. Consequently, B-Rep operators in CAD modelers are only binary operators and, during a design process, the user-selected one combines the latest primitive generated to the existing shape of M at the stage t of this generative process. Additionally, CAD modelers providing regularized Boolean operators reduce them to binary operators, even though they are n-ary ones, as classically defined in the CSG (Constructive Solid Geometry) approaches [31]. Here, we don't make any restriction on the amount of primitives possibly generated 'in parallel', i.e. the arity of the combination operators is $n \geq 2$.

The number of possible generative processes producing M can be arbitrary large, e.g. even a cube can be obtained from an arbitrary large number of extrusions of arbitrary small extent combined together with a union operator. We therefore refer to the concept of maximal primitives so that the number of primitives is finite and as small as possible for generating M.

A valid primitive P_i identified at a stage t using a base face F_{b_1} is said to be *maximal* when no other valid primitive P_j at that stage having F'_{b_1} as base face can be entirely inserted in P_i (see Section 4.1 and Figure 8a): $\forall P_j, P_j \not\subset P_i$. F_{b_1} is a maximal face as defined at Section 3.2.

Maximal primitives imply that the contour of a sketch can be arbitrary complex, which is not the case in current engineering practice, where the use of simple primitives eases the interactive modeling process, the parameterization, and geometric constraint assignments to contours. The concept of maximal primitives is analog to the concept of maximal volume used in [2,1] but [12] does not use it. Even if making use of maximal primitives considerably reduces the number of possible generative processes, they are far from being unique for M. We therefore make the further hypothesis that the generative processes we are looking for are principally of type additive, i.e. they are purely based on a regularized union operator when combining primitives at each stage t of generative modeling processes. This hypothesis is particularly advantageous when intending to tailor a set of generative processes that best fit the needs of idealization processes. Indeed, idealized structures, such as mid-surfaces, lie inside such primitives, and connections between primitives locate also the connections between their idealized representatives. Therefore, the idealized representation of M can be essentially derived from each P_i and its connections, independently of the other primitives in case of additive combinations. Figure 3a gives an example where M can be decomposed into two primitives combined with a union (b). M in Figure 3b can thus be idealized directly from these two primitives and their interface. On the contrary, when allowing material removal, idealization transformations are more complex to process, while the resulting volume shapes are identical. Figure 3c shows two primitives which, combined by Boolean subtraction, result also in object (a). However, computing an idealization of (a) by combining idealizations of its primitives in (c) is not possible. Performing the idealization of M from its primitives strengthens this process compared to [19–21] for two reasons: Firstly, each P_i and its connections bound the 3D location and the connections of other idealized primitives. Secondly, different categories of connections can be defined, which is important because idealization processes still rely on users' know-how to process connections significantly differing from reference ones (see Figure 13).

To further reduce the number of possible generative processes, the processes described should be non trivial variants of processes already identified. For example, the same rectangular block can be extruded with three different face contours and directions but they create the same volume. These equivalent processes can be detected when comparing the geometric properties of the contour. Other similar observations will be addressed in the following sections when describing the criteria to select meaningful generative processes.

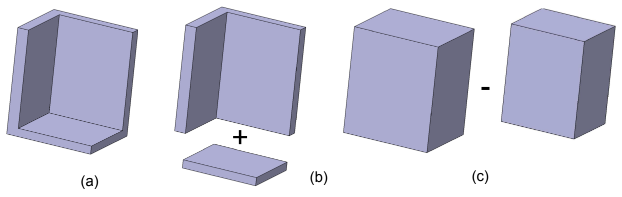


Fig. 3. a) Simple shape with idealizable sub domains, b) Primitives to obtain (a) with an additive process, c) Primitives to obtain (a) with a removal process.

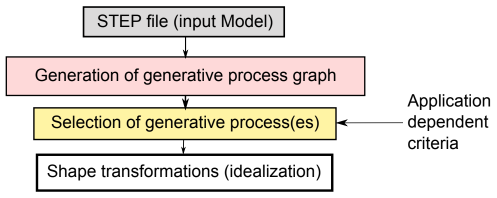

STEP file (input Model)
Generation of generative process graph
Selection of generative process(es)
Application dependent criteria
Shape transformations (idealization)

Fig. 4. Pipeline producing and exploiting generative shape processes.

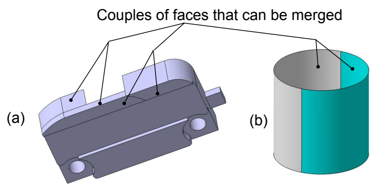


Fig. 5. Examples of configurations where faces must be merged to produce a unique boundary decomposition. a) face decomposition due to the modeling process, b) face decomposition due to topological requirements.

The above hypotheses aim at reducing the number of generative processes producing the same object M while containing primitives suited to idealization transformations independently of the design process initially set up by engineers.

The overall approach can be synthesized through the process flow of Figure 4. The input STEP file contains the B-Rep model M. A set of generative processes is extracted that form sets of construction trees, possibly producing a graph. To this end, application dependent criteria are used to identify one or more construction trees depending on the application needs. Here, we focus on criteria related to idealization for FEA.

3.2. *Intrinsic boundary decomposition using maximal entities*

In order to extract generative processes from the B-Rep decomposition of M, it is important to have a decomposition of M that uniquely characterizes its shape. A B-Rep decomposition of an object is however not unique (and thus not suitable), because it is subjected to two influences:

– its modeling process, whether it is addressed forward during a design process or backward as in the present work. Indeed, each operation involving a primitive splits/joins boundary faces and edges. When joining adjacent faces or edges, their corresponding surfaces or curves can be identical. Their decomposition is thus not unique. However, CAD modelers may not merge the corresponding entities, thus producing a boundary decomposition that is not changing the object shape (see Figure 5a). For our purposes, such configurations of faces and edges must lead to a merging process so that the object boundary decomposition is unique for a given shape;
– the necessary topological conditions to setup a consistent paving of an object boundary, i.e. the boundary decomposition must be a CW-complex. Consequently, curved surfaces need to be partitioned. As an example, a cylinder is decomposed into two half cylinders in most CAD modelers or is described with a self connected patch sewed along a generatrix (see Figure 5b). In either case, the edge(s) connecting the cylindrical patches are adjacent to the same cylindrical surface and are not meaningful from a shape point of view. Hence, for our purposes, they must not participate to the intrinsic boundary decomposition of the object.

Following these observations, the concepts of *maximal faces* and *edges* are introduced as a means to produce an intrinsic and unique boundary decomposition for a given object M. Maximal faces are identified first. For each face of M, a maximal face F is obtained by repeatedly merging an adjacent face F_a sharing a common edge with F when F_a is a surface of same type and same parameters than F. F is maximal when no more face F_a can be merged with F. Indeed, maximal faces coincide with 'c-faces' defined in [32] that have been proved to uniquely defined M. Similarly, for each edge of M, a maximal edge E with adjacent faces F_1 and F_2 is obtained by repeatedly merging an adjacent edge E_a when E_a is also adjacent to F_1 and F_2. Again, E is maximal when no more edge E_a can be merged with E. As a consequence of these merging processes, it is possible to end up with closed edges having no vertex or with closed faces having no edge. An example for the first case is obtained when generating the maximal face of the cylinder in Figure 5b. A sphere described with a single face without any edge and vertex is an example for the second case.

Because of maximal edges without vertices and faces without edges, merging operations are performed topologically only, i.e. the object's B-Rep representation is left unchanged. Maximal faces and edges are generated not only for the initial model M but also after the removal of each primitive when identifying the graph of generative processes. Consequently, maximal primitives are based on maximal faces and edges even if not explicitly mentioned throughout this document.

4. Generative processes

Given an input volume object, the first step of our method is to transform it into a blending radii free object M. For this, we apply defeaturing functions available in most CAD systems. This operation is a consequence of the modeling context defined in Section 3. Even though

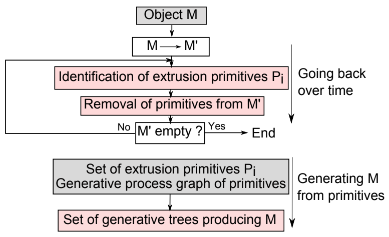

Object M
M ⟶ M'
Identification of extrusion primitives Pi
Removal of primitives from M'
No
M' empty ?
Yes
End
Going back over time
Set of extrusion primitives Pi
Generative process graph of primitives
Set of generative trees producing M
Generating M from primitives

Fig. 6. Overall scheme to obtain generative processes.

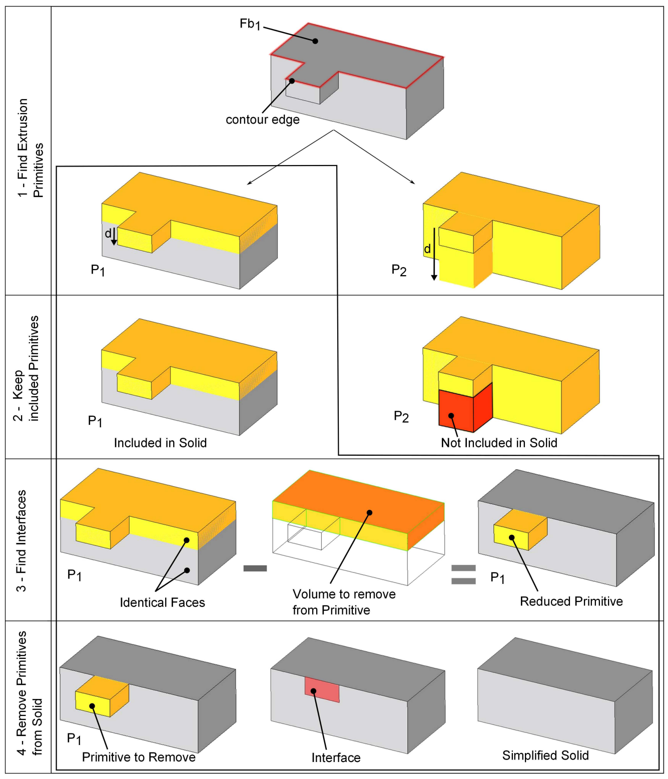


Fig. 7. An example illustrating the major steps for identifying a primitive P_i and removing it from the current model M_{-j}.

these functions may not be sufficient and robust enough, this is currently our working configuration. In contrast to blending radii, most chamfers are included in the present approach because they can be part of extrusion primitives and hence, included in the sketched contours used to define extrusion primitives. Even if CAD softwares provide specific functions for chamfers, they are devoted to the design context but basic operators of extrusion with material addition or removal could produce the same result, in general. This analysis regarding chamfers shows the effect of the concept of maximal primitives described at section 3.1.

Starting with the object M, the generative processes are obtained through two phases:

- M is processed by iterative identification and removal of primitives. The objective of this phase is to 'go back in time' until reaching root primitives for generative processes. The result of this phase is a set of primitives;
- based on hypotheses of section 3.1, a set of generative processes is produced using the primitives obtained at the end of the first phase to meet the requirements of an application: here idealization.

Figure 6 summarizes the overall scheme just described previously. Figure 7 illustrates the major steps of the extraction of a generative process graph, i.e. from the primitive identification up to its removal from M, and will be further explained in Sections 4.1 and 4.2.

4.1. *Extrusion primitives, visibility and attachment in M_{-j}*

In order to identify extrusion primitives P_i in $M = M_0$ and evolutions M_{-j} of it, backward at the j^{th} step of the generation of the generative process graph, it is mandatory to define its geometric parameters as well as the hypotheses taken in the present work (see Figure 8). First of all, notice that a 'reference primitive' P_i is never appearing entirely in M or M_{-j} unless it is isolated like a root of a construction tree, i.e. $P_i = M$ or $P_i = M_{-j}$. Apart from these particular cases, P_i are only partly visible. For simplicity, we refer to such P_i as 'visible primitives'. P_i is the memory of a generative process that took place between M_{-j} and $M_{-(j+1)}$. Extracting P_i significantly differs compared to feature recognition approaches [20,3,2,21,12] that use P_i and their neighborhood to subdivide the object. Here, identifying visible primitives enables the generation of reference ones having simpler contours.

The parameters involved in a reference extrusion P_i are the two base faces, Fb_1 and Fb_2, that are planar and contain the same sketched contour where the extrusion takes place. Considering extrusions that add volume to a pre-existing object, the edges of Fb_i are called *contour edges* which are convex. A convex edge is such that the normals at its adjacent faces define an angle α such that: $0 < \alpha < \pi$. When P_i belongs to M_{-j}, the contour edges along which P_i is attached to M_{-j} can be either convex or concave depending on the neighborhood of P_i in M_{-j} (see Figure 8a).

In the direction $\mathbf{d}$ of the extrusion, all the edges are straight line segments parallel to each other and orthogonal to Fb_i. These edges are named *lateral edges*. Faces adjacent to Fb_i are called *lateral faces*. They are bounded by four edges, two of them being lateral edges. Lateral edges can be *fictive lateral edges* when a lateral face coincides with a face of M_{-j} adjacent to P_i (see Figure 8a). When lateral faces of P_i coincide with adjacent faces in M_{-j}, there cannot be edges separating P_i from $M_{-(j+1)}$ because of the definition of maximal faces. Such a configuration refers to *fictive base edges* (see Figure 7 with the definition of primitive P_1).

Visibility. The *visibility* of P_i depends on its insertion in M_{-j} and sets the conditions to identify P_i in ∂M_{-j}. The simplest visibility is obtained when P_i's base faces Fb_i in M_{-j} exist and when at least one lateral edge connects Fb_i in M_{-j} (see Figure 8a).

More generally, P_i is identified using two conditions. First, at least one base face Fb_i is visible in M_{-j}, i.e. P_i can be identified by any planar maximal face. Second, one lateral edge exists, i.e. a straight line segment orthogonal and connected to a convex edge of Fb_i. This edge defines

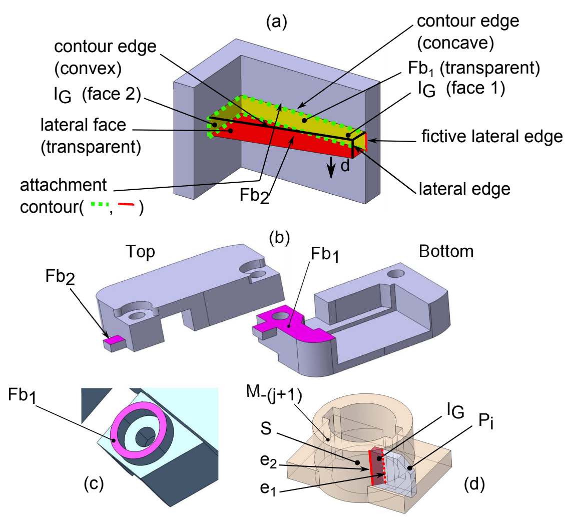


Fig. 8. a) Entities involved in an extrusion primitive. Visible extrusion feature with its two identical base faces Fb_1 and Fb_2. b) Visible extrusion feature with its two different base faces Fb_1 and Fb_2. c) Visible extrusion feature with a unique base face Fb_1 (detail of Figure 1a - 34B). d) Example of geometric interface I_G of type volume between P_i and $M_{-(j+1)}$.

the extrusion distance of P_i. Further lateral faces are identified by propagation from both sides of the candidate edge until the convexity of the edge of Fb_i no longer holds. Notice that the lateral edges mentioned may not be maximal edges when lateral faces are cylindrical because maximal faces may remove all B-Rep edges along a cylindrical area. These conditions of definition of extrusion distance restricts the range of extrusion primitives addressed compared to the use of the longest lateral segment existing in the lateral faces attached to Fb_i. However, it is a first step enabling to address a fair set of mechanical components and validate the major concepts of the proposed approach. This generalization is left for future work. Figure 8b, c give examples involving two or one visible base face, respectively.

Attachment. An extrusion primitive P_i is *attached* to M_{-j} in accordance to its visibility in M_{-j}. The attachment defines a geometric interface, I_G, between P_i and $M_{-(j+1)}$, i.e. $I_G = P_i \cap M_{-(j+1)}$. This interface can be a surface or a volume or both, i.e. a non-manifold model. One of the simplest attachments occurs when P_i has its base faces Fb_1 and Fb_2 visible with similar contours. This means that P_i is connected to $M_{-(j+1)}$ through lateral faces only. Consequently, I_G is a surface defined by the set of lateral faces not visible in P_i. Figure 8a illustrates such a type of interface (I_G contains two faces depicted in yellow).

A simple example of attachment involving a volume interface I_G between P_i and $M_{-(j+1)}$ is given in Figure 8d. Notice that the interface between P_i and $M_{-(j+1)}$ contains also a surface interface that is not highlighted. However, as we will see in Section 5, all possible variants of I_G must be evaluated to keep the acceptable ones.

In a first step, P_i can be translated directly into an algorithm to identify them (procedure *find_visible_extrusion* of algorithm 1). The visibility of P_i does not refer to its neighboring faces in M_{-j}. Next, they are subjected to validity conditions described in the following section.

4.2. *Primitive removal to go back in time*

The purpose is now to describe the removal operator that produces a new model $M_{-(j+1)}$ anterior to M_{-j}. This removal operator is defined as a binary operator with P_i and M_{-j} as operands and $M_{-(j+1)}$ as result. In the context of a generative process, M_{-j} relates to a step j and $M_{-(j+1)}$ to a step $(j+1)$.

Characterization of interfaces. In order to be able to generate $M_{-(j+1)}$ once P_i is identified, it is necessary to reconstruct faces adjacent to P_i so that $M_{-(j+1)}$ defines a volume. To this end, the faces of M_{-j} adjacent to P_i and I_G must be characterized. Here, we consider that P_i is adjacent to other subsets of primitives through one edge at least. The removal operator depends on the type of I_G and two categories can be set up to simplify the presentation of this operation:

1- I_G is of surface type. In this category, the removal operator will have to create lateral faces and/or the extension of Fb_2 so that the extended face coincides with Fb_1. Indeed, this category needs to be subdivided into two sub categories:
  a- I_G contains lateral faces of P_i only (see Figure 8a) or I_G contains also an extension of Fb_2 and edges of this extension are concave edges in $M_{-(j+1)}$;
  b- I_G may contains lateral faces of P_i but it contains an extension of Fb_2 and the edges of this extension are fictive base edges in M_{-j}. These edges would be convex edges in $M_{-(j+1)}$, (see P_1 in Figure 7);
2- I_G contains at least one volume sub domain.

In addition, considering that Fb_1 at least is visible and P_i is also visible (see section 4.1), the attachment contour may not be entirely available to form one or more edge loops (see Figure 8a). Also, I_G can contain more than one connected component when P_i is resembling a handle connected to $M_{-(j+1)}$, which produces more than one edge loop to describe the attachment of P_i to $M_{-(j+1)}$ in I_G.

Validity. Whatever the category of interface, once P_i is identified and its parameters are set (contour and extrusion distance), it is necessary to validate it prior to define its interface (step 2 of Figure 7). Let P_i designates the volume of the reference primitive, i.e. the entire extrusion P_i. To ensure that P_i is indeed a primitive of M_{-j}, the necessary condition is formally expressed using regularized Boolean operators between these two volumes:

$$(M_{-j} \cup^* P_i) -^* M_{-j} = \phi. \qquad (1)$$

This equation states that P_i intersects M_{-j} only along the edge loops forming its attachment to $M_{-(j+1)}$, i.e. P_i does not cross the boundary of M_{-j} at other location than its attachment. The regularized Boolean subtraction states that limit configurations producing common points , curve segments or surface areas between P_i and M_{-j} at other locations than the attachment of P_i are acceptable. This condition strongly reduces the number of primitives over time. Figure 7 at step 2 shows that primitives P_2 and P_3 can be discarded.

Removal of P_i. The next step is to generate $M_{-(j+1)}$ once P_i has been identified and removed from M_{-j}. Depending of the type of I_G, some faces of P_i may be added to ensure that $M_{-(j+1)}$ is a volume.

• If I_G is of type 1a, then the faces adjacent to the contour edges of Fb_1 are orthogonal to Fb_1. These faces are either planar or cylindrical. I_G contains the faces extending these faces, F_{a_1}, to form the lateral faces of P_i that were 'hidden in M_{-j}'. Edges of the attachment of P_i belonging to lateral faces of P_i can be lateral edges (either real or fictive ones) or arbitrary ones. Lateral edges bound faces in F_{a_1}, arbitrary edges bound the extension of the partly visible lateral faces of P_i, they belong to: F_{a_2}. Then, I_G may contain the extension of Fb_2 called F_{a_3} such that: $Fb_2 \cup F_{a_3} = Fb_1$. Then:

$$\partial M_{-(j+1)} = (\partial M_{-j} - \partial P_i) \cup (F_{a_1} \cup F_{a_2} \cup F_{a_3}), \qquad (2)$$

where ∂M_{-j} is the set of connected faces bounding M_{-j}, ∂P_i is the set of connected faces bounding the visible part of P_i. $\partial M_{-(j+1)}$ defines a closed, orientable surface, without self intersection. $M_{-(j+1)}$ is therefore a volume.

• If I_G is of type 1b, I_G contains a set of faces extending lateral faces of P_i: F_{a_1}. To reduce the description of the various configurations, let us focus on the key aspect related to the extension of Fb_2 contained in I_G. If this extension can be defined like F_{a_3} above, it has to be observed that fictive edges of this extension in M_{-j} are replaced by convex edges in $M_{-(j+1)}$, i.e. edges of the same type (convex) as their corresponding edges in Fb_1 (see Figure 7 step 3 left image). Without going into details, these fictive edges can be removed to simplify the contour of P_i since they bring unnecessary complexity to P_i and does not affect the complexity of $M_{-(j+1)}$. The corresponding effect is illustrated on Figure 7 steps 3 and 4. This contour simplification can influence the contents of the sets F_{a_1} and F_{a_3} above but it has no impact on the integrity of the volume $M_{-(j+1)}$ obtained.

• If I_G belongs to category 2, it contains at least one volume sub domain. Here again the diversity of configurations can be rather large and it is not intended to give a detailed description of this category. A first condition to generate a volume interface relates to surfaces adjacent to P_i. If S is the extension of such a surface and $S \cap^* P_i \neq \phi$, S may contribute to the generation of a volume sub domain. Then, each of these surfaces has to be processed. To this end, all the edges attaching P_i in $M_{-(j+1)}$ and bounding the same surface in $M_{-(j+1)}$ are grouped together since they form a subset of the contour of faces possibly contributing to a volume sub domain. These groups are named E_a. Such an example of edge grouping is given in Figure 8d where e_1 and e_2 are grouped because of their adjacency between P_i and the same cylindrical surface. E_a, together with other sets of edges are used to identify loops in S that define a volume sub domain of I_G that must satisfy validity conditions not described here for sake of conciseness.

There may be several valid volume sub domains defining alternative sets of faces to replace the visible part of P_i, ∂P_i, in ∂M_{-j} by sets of faces that promote either the extension of surfaces adjacent to P_i or the imprint of P_i in $M_{-(j+1)}$ with the use of faces belonging to the hidden part of P_i in M_{-j}. All the variants are processed to evaluate their possible contribution to the generative process graph.

If, in a general setting, there may be several variants of I_G to define $M_{-(j+1)}$, these variants always produce a realisable volume, which differs from the halfspace decomposition approaches studied in [23,24] where complement to the halfspaces derived from their initial boundary were needed to produce a realisable volume.

5. Extracting the generative process graph

5.1. *Filtering out the generative processes*

Having defined the primitive removal operator, the purpose is now to incorporate constraints on variants of I_G so that a meaningful set of models M_{-j}, $j > 0$, can be generated to produce a generative process graph. As mentioned earlier, the purpose is to 'go back in time' from model M to single primitives forming the roots of possible construction trees. To this end, any acceptable primitive removal at step j of the graph generation must produce a transformation of M_{-j} into k objects $M_{-(j+1)_k}$ using I_{G_k}, one of the variants of I_G, such that $M_{-(j+1)_k}$ are simpler than M_{-j}. This simplicity concept is a necessary condition for the graph generation to converge toward a set of construction trees having a single primitive as root. Consequently, the simplicity concept applied to the transition between M_{-j} and $M_{-(j+1)_k}$ is sufficient to ensure the convergence of the graph generation process.

The shape simplification occurring between M_{-j} and $M_{-(j+1)_k}$ can be defined as follows. First of all, it has to be considered that ∂M_{-j} and $\partial M_{-(j+1)_k}$ contain maximal faces and edges. In fact, after P_i is removed and replaced by I_{G_k} to produce $M_{-(j+1)_k}$, its boundary decomposition is re-evaluated to contain maximal faces and edges only. Then, let n_j be the number of (maximal) faces in M_{-j} and $n_{(j+1)_k}$ be the same quantity for $M_{-(j+1)_k}$, the quantity δ_{jk}: $\delta_{jk} = n_j - n_{(j+1)_k}$ characterizes the shape simplification under the variant I_{G_k} if $\delta_{jk} \geq 0$. This condition is justified because it enforces a 'diminishing number of maximal faces over time', which is an intrinsic quantity to each shape.

5.2. *Generative process graph*

Having defined the condition to evolve backward in the generative process graph, the graph generation is summarized with algorithm 1. The main procedure *Extract_graph* processes the *node_list* current variants of the model at the current step 'backward in time' using the procedure

Algorithm 1 Extract generative process graph

```
procedure  $Extract\_graph$
    $input M$
    $node\_list \leftarrow root$; $current\_node \leftarrow root$;
    $arc\_list \leftarrow nil$; $current\_arc \leftarrow nil$; $node\_list(0) = M$
    while $size(node\_list) > 0$ do
        $current\_node = last\_element\_of\_list(node\_list)$
        $M_{-j} = get\_solid(current\_node)$
        $config\_list = Process\_variant(M_{-j})$
        $compare\_config(get\_all\_config(graph), config\_list)$
        for each $config$ in $config\_list$ do
            $M_{-(j+1)} = remove\_primitives(M_{-j}, config)$
            $node = generate\_node(M_{-(j+1)}, config)$
            $add\_node(graph, node)$
            $arc = generate\_arc(node, current\_node)$
            $add\_arc(graph, arc)$
            $append(node\_list, node)$
        $remove\_element\_from\_list(node\_list, current\_node)$
procedure  $config\_list = Process\_variant(M_{-j})$
    $initialize\_primitive\_list(prim\_list)$
    $ext\_list = find\_extrusion(M_{-j})$
    for  each $P_i$ in $ext\_list$ do
        $P_i = simplify\_prim\_contour(P_i, M_{-j})$
        $interf\_list = generate\_geom\_interfaces(P_i, M_{-j})$
        $interf\_list = discard\_complex(interf\_list, P_i, M_{-j})$
        if $size(interf\_list) = 0$ then
            $remove\_from\_list(P_i, ext\_list)$;
        $append(prim\_list, interf\_list(i))$
    $sort\_primitive(prim\_list)$
    $config\_list = generate\_independent\_ext(prim\_list, M_{-j},)$
procedure $ext\_list = find\_extrusion(M_{-j})$
    $ext\_list = find\_visible\_extrusions(M_{-j})$;
    $ext\_list = remove\_ext\_outside\_model(M_{-j}, ext\_list)$;
    $ext\_list = remove\_ext\_included\_ext(ext\_list)$;
```

Process_variant and compares the new variants to the existing graph nodes using *compare_config*. If variants are identical, graph nodes are merged, which creates cycles. Then, *Extract_graph* adds a tree structure to a given variant corresponding to the new simpler variants derived from M_{-j}. The graph is completed when there is no more variant to process, i.e. *node_list* is empty. Here, the purpose is to remove (using *remove_primitives*) the largest possible amount of primitives P_i whose interfaces I_{G_k} are not overlapping each other, i.e. $\forall(i,j,k,l), i \neq j$, $I_{G_k} \in P_i$, $I_{G_l} \in P_j$, $I_{G_l} \cap I_{G_k} = \phi$, otherwise δ_{jk} would not be meaningful. Selecting the largest possible amount of P_i and assigning them to a graph node is mandatory to produce a compact graph. Each such node expresses the fact that all its P_i could be removed, one by one, in an arbitrary order, which avoids describing trivial ordering changes.

To process each variant M_{-j} of M, *Process_variant* starts with the identification of valid visible extrusion primitives in M_{-j} using *find_extrusion* (see Sections 4.1 and 4.2 respectively). However, to produce maximal primitives, all valid primitives which can be included into others (because their contour or their extrusion distance is smaller than the others) are removed (*remove_ext_included_ext*).

Once valid maximal primitives have been identified, processing the current variant M_{-j} carries on with contour simplification: *simplify_prim_contour*, if it does not impact the shape complexity of $M_{-(j+1)}$ (see Section 4.2). Then, all the valid geometric interfaces I_{G_k} of each primitive are generated with *generate_geom_interfaces* (see Section 4.2) and interfaces I_{G_k} increasing the shape complexity are discarded with *discard_complex* to ensure the convergence (see Section 5.1). Sets of independent primitives are ordered to ease the user's navigation in the graph.

5.3. *Results of generative process graph extraction*

The previous algorithm has been applied to a set of components whose shapes are compatible with extrusion processes to stay consistent with algorithm 1 though they are industrial components. The results have been obtained automatically using algorithm 1 implemented using Python and bindings with Open Cascade (OCC) library. Statistics given are the amount of calls to a generic Boolean type operator available in the OCC library, the total number of visible primitives (*find_visible_extrusions*), n_v, and the final number of P_i in the graph, n_p.

Figure 9 shows the generative processes extracted from four different and rather simple components. They are characterized by triples $(n_B; n_v; n_p)$, (2183; 220; 8), (8246; 225; 15), (1544; 132; 6), (9353; 240; 31), for a, b, c and d, respectively. The graph structure reduces to a tree one for each of them. It shows that merging all extrusions in parallel into a single node can be achieved and results into a compact representation. These results also show the need for a constraint, that we can formalize as follows: configurations produced by *generate_independent_ext* must be such that each variant $M_{-(j+1)_k}$ generated from M_{-j} must contain a unique connected component as it is with M. However, this has not been implemented yet. This continuity constraint expresses the fact that M is a continuous medium and its design process follows this concept too. Consequently, any of its transformation stages must be so to ensure that any simplified model, i.e. any graph node, can stand as basis for an idealization process. Then, it is up to the idealizations and their hypotheses to remove such a constraint, e.g. when replacing a primitive by kinematic boundary conditions to express a rigid body behavior locally.

Figure 10 shows the graph extracted from the component analyzed in Figure 1. It is characterized by (111789; 1440; 62). Two variants appear at step 4 and later merge at step 8. It effectively produces a graph structure. It can be observed that the construction histories are easier to understand for a user than the one effectively used to model the object (see Figure 1). Clearly, the extrusion primitives better meet the requirements of an idealization process and they are also better suited to dimension modification processes as mentioned in Section 1. It has been shown that this graph is a promising basis for getting a better insight of a shape structure and evaluate its adequacy for idealizations.

6. Performing idealizations from a generative process graph

Here, we briefly illustrate how a generative process graph obtained with algorithm 1 can be used in shape idealizations.

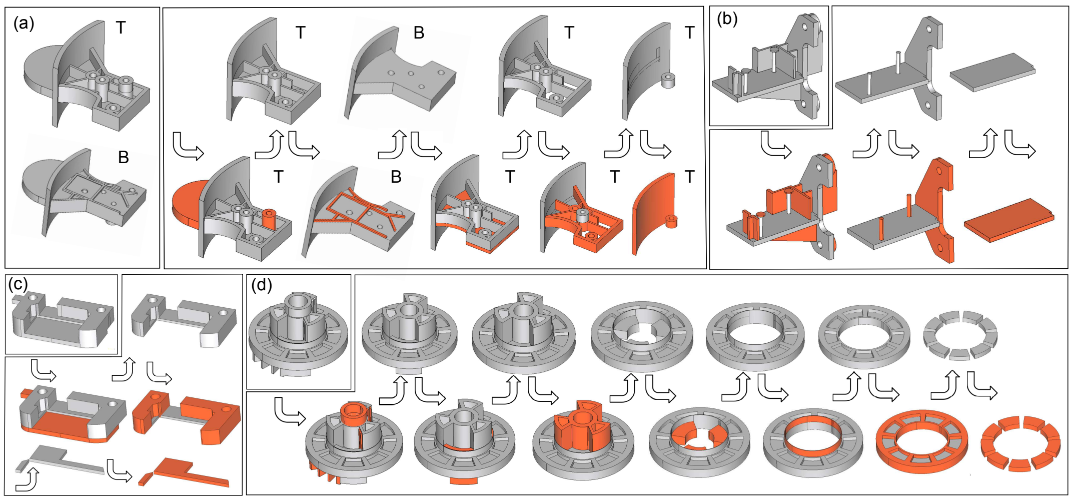


Fig. 9. Extraction of generative processes for four different components: a, b, c, d. Orange sub domains highlight the set of visible primitives removed at each step of the graph generation. Construction graph reduces to a tree for each of these components. a) T and B indicate Top and Bottom views to locate easily the primitives removed. Other components use a single view.

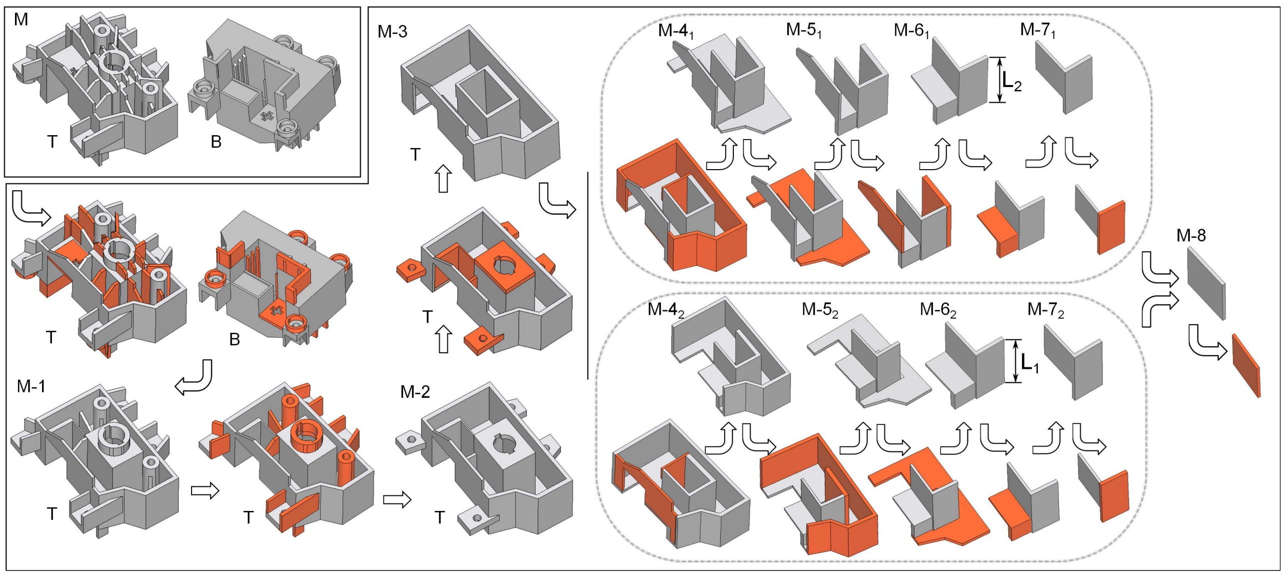


Fig. 10. Generative process graph of component analyzed on Figure 1. T and B convey the same meaning. Orange sub domains indicate the removed primitives at each graph node. Label M_{-j_k} indicates the step number j when 'going back in time' and the existence of variants k, if any. Steps M_{-6_1} and M_{-6_2} differ because lengths L_1 and L_2 are not identical.

6.1. *Evaluating primitives for idealization*

The primitives extracted from the graph can be used to analyze their morphology and evaluate their adequacy for idealizations. Because the primitives are all extrusions and adding material, analyzing their morphology can be performed with the MAT [4,25,21]. MAT is particularly suited to extrusion primitives since it can be applied in 2D because of the constant thickness of P_i and can be used to decide whether P_i can be assigned a plate or shell mechanical behavior. In the present case, the P_i obtained lead to two distinct configurations (see Figure 11). Figure 11a shows a configuration with a thin extrusion, i.e. the maximal diameter Φ obtained with the MAT from the P_i's contour is much larger than P_i's thickness defined by the extrusion distance d. Then, the dimensional reduction of P_i would be a surface parallel to the base face having P_i's contour. Figure 11b shows a configuration where the morphology of P_i leads to an idealization that would be based on the content of the MAT because d is much larger than Φ.

To idealize a sub domain of an object in mechanics, a reference proportion used to decide whether it is idealizable or not is a ratio of ten between its in-plane dimensions and its thickness. This can be formalized as: $x = \max((\max\Phi/d), (d/\max\Phi))$ and x is applicable for all morphologies of P_is. Another threshold can be user-defined to tune the morphological analysis and decide when P_i can/cannot be idealized. Figure 11c illustrates a configuration where the morphological analysis does not produce a ratio $x > 10$ though a user might idealize P_i as a plate. Let

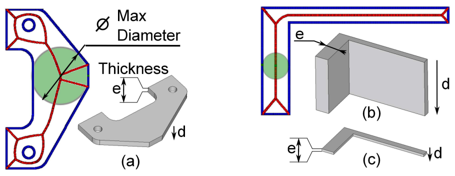


Fig. 11. Indication of idealization direction of extrusion primitives with 2D MAT applied their contour.

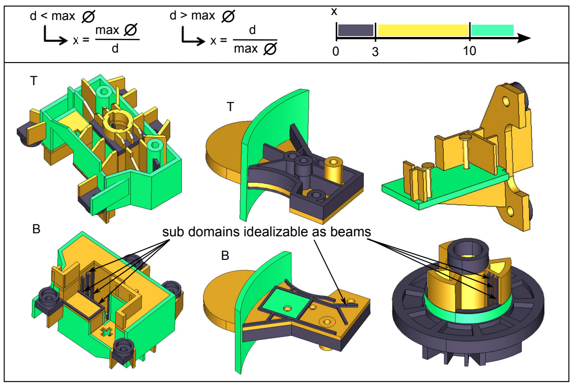


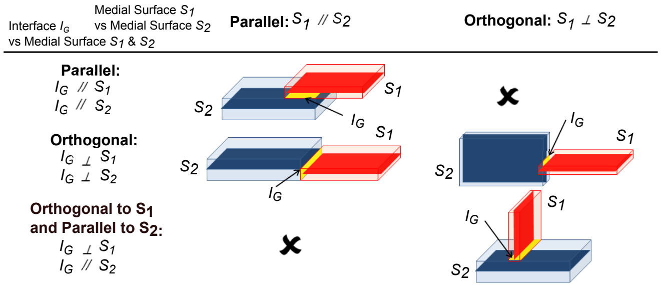
Fig. 12. Idealization analysis of components taken from Figures 9 and 10 decomposed into a set of extrusion primitives using their construction graph. Violet indicates sub domains that cannot be idealized as plates or shells, green ones can be idealized and yellow ones can be subjected to user decision.

$x = 3$ be this user-defined value, Figure 12 shows the result of the interactive analysis the user can perform from the decomposition results obtained in Figures 9 and 10. Colors interpretation is given in the figure caption. The analysis is applied to the graph rather than a single tree structure to improve the efficiency of the analysis. However, the result obtained on component of Figure 10 shows that the variants in the construction graph have no influence with respect to the morphological analysis criterion, in the present case. Results on components of Figure 10 and 9a show the limit of this criterion because some non-idealizable P_i (see indications on Figure 12 regarding violet sub domains) are indeed well proportioned to be idealized with beams. Such configurations are clearly calling for complementary criteria.

6.2. *Processing connections between idealized primitives*

If the morphological analysis of P_i is one application of the generative process graph, the primitive decomposition obtained can be used to monitor the idealizations. To this end, a taxonomy of connections between extrusion primitives is summarized in Figure 13. The taxonomy refers to parallel and orthogonal configurations for simplicity but these configurations can be extended to process a larger range of angles. More specifically, the configuration where I_G is orthogonal to S_1 and S_2 both is lacking of robust solutions [20,21] and other connections can require deviation from mid-surface location.

Having decomposed M into extrusion primitives P_i, the



Interface I_G vs Medial Surface S_1 & S_2
Medial Surface S_1 vs Medial Surface S_2
Parallel: S_1 // S_2
Orthogonal: $S_1 \perp S_2$
Parallel: I_G // S_1 I_G // S_2
Orthogonal: $I_G \perp S_1$ $I_G \perp S_2$
Orthogonal to S_1 and Parallel to S_2: $I_G \perp S_1$ I_G // S_2
S_1
S_2
I_G

Fig. 13. Taxonomy of connections between extrusion primitives.

location of interfaces I_G between P_i are precisely identified and can be used to monitor the deviations needed from mid-surfaces to improve the idealization process and take into account the user's know-how. Particularly, connections with parallel mid-surfaces can be handled with mid-surface repositioning (see P_1 and P_2 on Figure 14b) and a corresponding adjustment of the material thickness on both sides of the idealized surface. This is a current practice in linear analysis that has been advantageously implemented using the relative position of extrusions. Similarly, when S_1 and S_2 are orthogonal to each other and their I_G is located at their boundary (see P_2 and P_3 on Figure 14b), either of the mid-surfaces needs to be relocated to avoid meshing narrow areas along one of the P_i's boundaries (here P_3 is moved according to d_3). Again, this configuration can be processed using the precise location of I_G.

6.3. *Results of idealization processes*

Figure 14a illustrates a component with its decomposition through the generative process graph and the corresponding interfaces between its extrusion primitives. This decomposition contains a set of primitive connections of both categories discussed in Section 6.2 and Figure 14b shows the repositioning of mid-surfaces among P_1, P_2 and P_3 that improves their connections and the overall idealization process. Figure 14c shows the resulting idealized model and its corresponding FE mesh.

7. Conclusion and future work

Construction trees and shape generation processes are common approaches to model mechanical components. Here, it has been shown that construction trees can be extracted from the B-Rep model of a component. Indeed, construction trees are structured into a graph to represent the non trivial collection of generative processes that produce the input B-Rep model. The graph contains non trivial construction trees in the sense that variants of extrusion directions producing the same primitive are not encoded, material addition operations that can be conducted in parallel are grouped into a single graph node to avoid the description of combinatorial combinations when primitives are added sequentially as in CAD software.

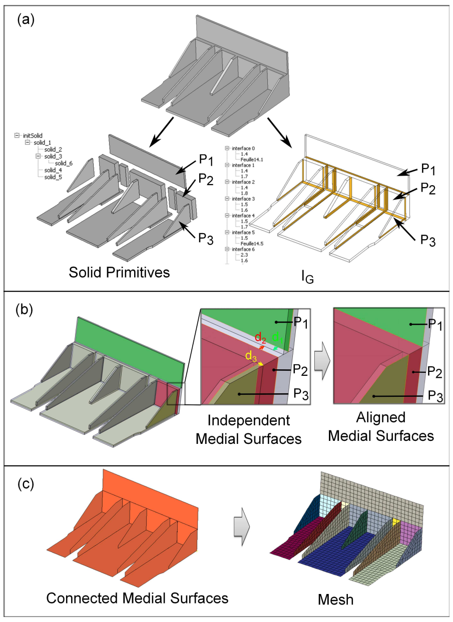


Fig. 14. Idealization process of a component taking advantage of its generative process graph, its corresponding primitives as well as the geometric interfaces between these primitives.

The benefit of a generative process graph has been evaluated in the context of idealizations as needed for FEA. Also, the object decomposition produces an accurate description of geometric interfaces between primitives, which has been advantageously used to set up idealizations.

Clearly, future work will focus on incorporating material removal operations and revolutions to extend the range of objects that can be processed.

Acknowledgments

This work is carried out in the framework of the ANR project ROMMA (RObust Mechanical Models for Assemblies) referenced ANR-09-COSI-012 and the authors thank the ANR for its financial support. This work is also supported by the ERC Expressive.

References

[1] Y. Woo, Fast cell-based decomposition and applications to solid modeling, CAD 35 (2003) 969–977.
[2] Y. Woo, H. Sakurai, Recognition of maximal features by volume decomposition, CAD 34 (2002) 195–207.
[3] Y. Lu, R. Gadh, T. J. Tautges, Feature based hex meshing methodology: feature recognition and volume decomposition, CAD 33 (3) (2001) 221–232.
[4] J. E. Makem, C. G. Armstrong, T. T. Robinson, Automatic decomposition and efficient semi-structured meshing of complex solids, in: Proceedings of the 20th International Meshing Roundtable, 2012, pp. 199–215.
[5] S.-S. Liu, R. Gadh, Automatic hexahedral mesh generation by recursive convex and swept volume decomposition, in: 6th International Meshing Roundtable, Sandia National Laboratories, 1997, pp. 217–231.
[6] M. Leyton, A Generative Theory of Shape, Lecture Notes in Computer Science LNCS 2145, Springer-Verlag, 2001.
[7] S. Gao, W. Zhao, H. Lin, F. Yang, X. Chen, Feature suppression based cad mesh model simplification, CAD 42 (12) (2010) 1178–1188.
[8] K. Y. Lee, C. G. Armstrong, M. A. Price, J. H. Lamont, A small feature suppression/unsuppression system for preparing b-rep models for analysis, in: Proceedings of the 2005 ACM symposium on Solid and physical modeling, SPM '05, 2005, pp. 113–124.
[9] J. H. Vandenbrande, A. G. Requicha, Spatial reasoning for the automatic recognition of machinable features in solid models, IEEE PAMI 15 (12) (1993) 1269–1285.
[10] S. Joshi, T. C. Chang, Graph-based heuristics for recognition of machined features from a 3d solid model, CAD 20 (2) (1988) 58–66.
[11] J. Han, M. Pratt, W. C. Regli, Manufacturing feature recognition from solid models: A status report, IEEE Transactions on Robotics and Automation 16 (2000) 782–796.
[12] K. Jha, B. Gurumoorthy, Multiple feature interpretations across domains, Computers in Industry 42 (2000) 13–32.
[13] J. Y. Lee, J.-H. Lee, H. Kim, H.-S. Kim, A cellular topology-based approach to generating progressive solid models from feature-centric models, CAD 36 (3) (2004) 217–229.
[14] S. Kim, K. Lee, T. Hong, M. Kim, M. Jung, Y. Song, An integrated approach to realize multi-resolution of b-rep model, in: Proceedings of the 2005 ACM symposium on Solid and physical modeling, SPM '05, 2005, pp. 153–162.
[15] J. Seo, Y. Song, S. Kim, K. Lee, Y. Choi, S. Chae, Wrap-around operation for multi-resolution cad model, CAD and Applications 2 (1-4) (2005) 67–76.
[16] H. L. Lockett, M. D. Guenov, Graph-based feature recognition for injection moulding based on a mid-surface approach, CAD 37 (2) (2005) 251–262.
[17] H. Zhu, C. H. Menq, B-rep model simplification by automatic fillet/round suppressing for efficient automatic feature recognition, CAD 34 (2) (2002) 109 – 123.
[18] S. Venkataraman, M. Sohoni, R. Rajadhyaksha, Removal of blends from boundary representation models, in: Proceedings of the seventh ACM symposium on Solid modeling and applications, SMA '02, 2002, pp. 83–94.
[19] C. S. Chong, A. S. Kumar, K. H. Lee, Automatic solid decomposition and reduction for non-manifold geometric model generation, CAD 36 (13) (2004) 1357–1369.
[20] M. Rezayat, Midsurface abstraction from 3d solid models: general theory and applications, CAD 28 (1) (1996) 905–915.
[21] D.-P. Sheen, T.-G. Son, D.-K. Myung, C. Ryu, S. H. Lee, K. Lee, T. J. Yeo, Transformation of a thin-walled solid model into a surface model via solid deflation, CAD 42 (8) (2010) 720–730.
[22] T. T. Robinson, C. Armstrong, R. Fairey, Automated mixed dimensional modelling from 2d and 3d cad models, Finite Elem. Anal. Des. 47 (2) (2011) 151–165.
[23] V. Shapiro, D. L. Vossler, Separation for boundary to csg conversion, ACM Trans. Graph. 12 (1) (1993) 35–55.
[24] S. F. Buchele, R. H. Crawford, Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations, CAD 36 (2004) 1063–1073.
[25] T. T. Robinson, C. G. Armstrong, G. McSparron, A. Quenardel, H. Ou, R. M. McKeag, Automated mixed dimensional modelling for the finite element analysis of swept and revolved cad features, in: Proceedings of the 2006 ACM symposium on Solid and physical modeling, SPM '06, 2006, pp. 117–128.
[26] J. Yang, S. Han, Repairing cad model errors based on the design history, CAD 38 (2006) 627–640.
[27] M. Li, F. C. Langbein, R. R. Martin, Constructing regularity feature trees for solid models, in: Proc. Geometric Modeling and Processing; LNCS, 2006, pp. 267–286.
[28] M. Li, F. Langbein, R. Martin, Detecting design intent in approximate cad models using symmetry, CAD 42 (2010) 183–201.

[29] M. Belaziz, A. Bouras, J.-M. Brun, Morphological analysis for product design, CAD 32 (5-6) (2000) 377–388.

[30] T. Lim, H. Medellin, C. Torres-Sanchez, J. R. Corney, J. M. Ritchie, J. B. C. Davies, Edge-based identification of dp-features on free-form solids, IEEE Trans. PAMI 27 (6).

[31] M. Mäntylä, An introduction to solid modeling, Computer Science Press, College Park, MD, 1988.

[32] C. E. Silva, Alternative definitions of faces in boundary representations of solid objects, Tech. rep., Carnegie Mellon University (1981).