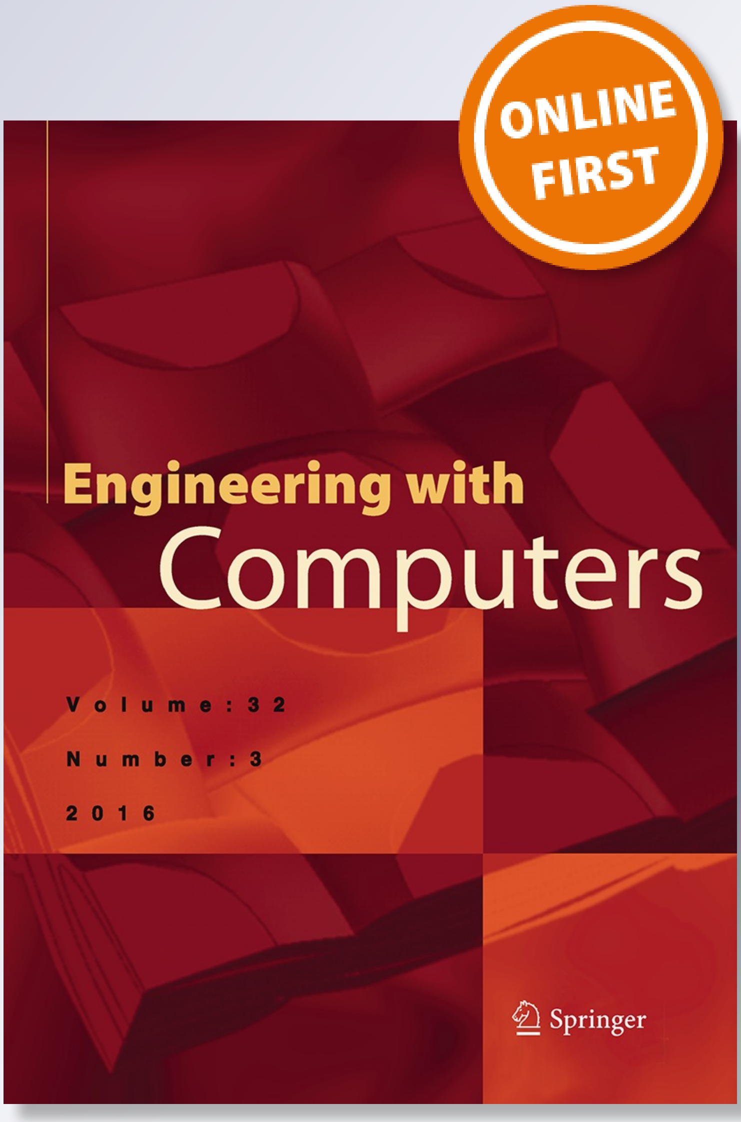
Leveraging feature generalization and decomposition to compute a well-connected midsurface

Yogesh Kulkarni

Engineering with Computers
An International Journal for Simulation-Based Engineering

ISSN 0177-0667

Engineering with Computers
DOI 10.1007/s00366-016-0466-x



Engineering with Computers
DOI 10.1007/s00366-016-0466-x

ORIGINAL ARTICLE

Leveraging feature generalization and decomposition to compute a well-connected midsurface

Yogesh Kulkarni[1]

Received: 25 November 2015 / Accepted: 1 July 2016

Abstract Computer-aided design (CAD) models of thin-walled parts, such as sheet metal or plastic parts, are often represented by their corresponding midsurfaces for computer-aided engineering (CAE) analysis. The reason being, 2D surface elements, such as "shell" elements, which need to be placed on the midsurface, provide fairly accurate results, while requiring far lesser computational resources time compared to the analysis using 3D solid elements. Existing approachesof midsurface computation are not reliable and robust. They result in ill-connected midsurfaces having missing patches, gaps, overlaps, etc. These errors need to be corrected, mostly by a manual and time-consuming process, requiring from hours to even days. Thus, an automatic and robust technique for computation of a well-connected midsurface is the need of the hour. This paper proposes an approachwhich, instead of working on the complex final solid shape, typically represented by B-rep (boundary representation), leverages feature information available in the modern CAD models for techniques such as defeaturing, generalization, and decomposition. Here, first, the model is defeaturedby removing irrelevant features, generating a simplified shape called "gross shape". The remaining features are then generalizedto their corresponding generic loft-feature equivalents. The model is then decomposed into sub-volumes, called "cells" having respective owner loft features. A graph is populated, with the cells at the graph nodes. The nodes are classified into midsurface patch-generating nodes (called 'solid cells' or *sCell*s) and interaction-resolving nodes (called 'interface cells' or *iCell*s). Using owner loft feature's parameters, *sCell*s compute their own midsurface patches. Using a generic logic, the patches then get connected appropriately in the *iCell*s, resulting in a well-connected midsurface. The efficacy of the approach is demonstrated by computing well-connected midsurfaces of various real-life sheet metal parts.

Keywords CAD · CAE · Features · Midsurface · Sheet metal features · Cellular decomposition

✉ Yogesh Kulkarni
kulkarniay@gmail.com

[1] MS College of Engineering Pune, Pune, Maharashtra, India

1 Introduction

Nowadays, CAE analysis is often performed at earlier stages in the digital product development process, to validate various design alternatives quickly. In such cases, the CAD models are often simplified, not only for quicker and fairly accurate results, but also to save on precious computing resources and time. Such simplifications involve mainly two phases—first, "defeaturing", which deals with removal of irrelevant features and, second, "dimension reduction" (also known as "idealization") in which solid shapes such as slender bar or thin-walled portions are represented by their lower-dimensional equivalents ([15]). The present paper focuses on one such idealization called "Midsurface", typically used for CAE analysis of thin-walled parts such as sheet metal and plastic parts [33].

Midsurface can be envisaged of as a surface, which lies midway of a thin-walled model, mimicking its shape. It is primarily used to place 2D shell elements in CAE analysis, which give comparable results compared tocomputationally expensive 3D solid element analysis. Due to this crucial advantage, midsurface functionality is available in many commercial CAD–CAE packages. Despite its huge demand, the existing approaches fail to compute a

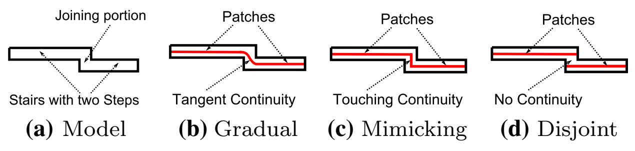


Fig. 1 Expectations vary with regard to the desired midsurface

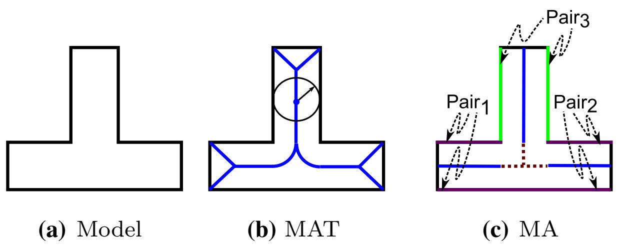


Fig. 2 Midsurface computation methods: MAT and MA

well-connected midsurface, especially in the case of complex models ([3, 26, 29, 35, 39]). Typical failures are missing midsurface patches, gaps, overlaps, not lying midway, not mimicking the input shape, etc. Correcting these errors is mostly a manual, tedious and highly time-consuming process, requiring several hours to days. This correction time can be nearly equivalent to the time it can take to create the midsurface manually from scratch ([34]). So, an automated and robust approach to compute midsurface is a critical need of the hour.

One of the major impediments in the development of such approach is the lack of precise definition of midsurface ([27]).

Different applications which use midsurface have different expectations about its shape, especially at the connection points. Figure 1 shows how midsurface expectation varies ([39]). Figure 1a shows a step-shaped thin-walled solid CAD model represented schematically as a 2D shape. Figure 1b shows two midsurface patches being joined in a gradual manner, i.e., with tangent geometric continuity (G_1). Such midsurface is preferred for CAE analysis. Figure 1c shows a planar patch being added to join the two midsurface patches, i.e., with touching continuity (G_0). This midsurface mimics the original model exactly and is best suited for shape matching or retrieval kind of applications where exact representation is preferred. Figure 1d shows the midsurface patches unconnected, i.e., with no continuity (0) and such output can be used where a disconnect needs to be highlighted. A closer observation will reveal that all the variations shown in Fig. 1 are due to the continuities between midsurface patches at the connections.

Thus, the goal of the present research work is to compute a midsurface that geometrically and topologically mimics the input shape i.e., Fig. 1c case.

2 Related work

Some of the prominent methods for computing the midsurface (generically called "medial") are thinning, chordal axis transform (CAT), parametric, medial axis transform (MAT), midsurface abstraction (MA), decomposition, etc. ([22]). Out of these, MAT and MA are two of the most researched medial computation techniques ([32]).

MAT is a locus of the center of an inscribed disc of maximal diameter as it rolls around the object's interior (Fig. 2b). It was initiated by Blum ([7]) and developed further into different variations ([1, 16, 17, 22, 27, 29]).

In MA (Fig. 2c), opposite faces are detected by ray casting, to form face pairs. Each face pair computes its own midsurface patch. Such patches are then "sewed" after extending and trimming, wherever necessary. Rezayat ([28]) initiated this approach and was developed further into various techniques ([10, 11, 14, 18, 19, 21, 23, 31, 32, 35, 39]).

Both MAT and MA have quite a few drawbacks.MAT creates unnecessary branches and its shape is smaller than the input shape. MA has two critical challenges, viz., detection of face pairs which create midsurface patches and joining of the midsurface patches.

Quite a few attempts using cellular decomposition for computing are found in the literature ([11, 12, 14, 39]).

Chong et al. [14] used concave edge decomposition, calculated midcurves by collapsing edge pairs and if they formed a loop created a midsurface patch. Their detection of edge pairs was with hard-coded values and the patch connection logic was not generic and comprehensive.

Cao et al. ([11, 12]) used concave edge decomposition and created midsurface patches using face pairs, but provided no elaboration on the joining method.

Woo used maximal volume decomposition ([38, 39, 41]), created midsurface patches using face pairs and joined them using union Boolean. Criteria for removing unwanted patches did not cater to all situations. Only analytical surfaces and parallel face pairs were handled in his work.

Boussuge et al. ([8]) used generative decomposition, recognized Extrude features from the cells, created midsurface patches in each cell and connected them together. This work was able to handle only additive Extrude features and connection types were limited to parallel and orthogonal types.

Zhu et al. ([43]) used virtual decomposition and created midsurface patches using face pairs. This research handled only three connection types.

After analyzing various past approaches, it can be concluded that there was limited success in computing a well-connected midsurface.

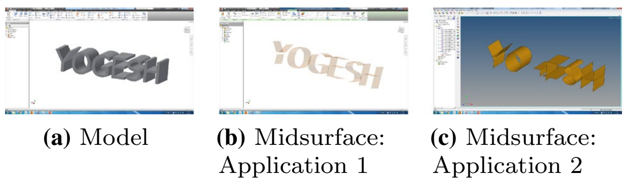


Fig. 3 Midsurface outputs of commercial applications

3 Motivation

Midsurface is the most suitable representation for the CAE analysis of thin-walled parts' CAD models. A Sandia report [25] states that, for the complex engineering models, their simplification amounts to about 60 % of the overall analysis time, whereas the mesh generation consumes about 20 % and solving the actual problem takes about 20 %. In the shipping industry, more than 80 % of CAE engineer's time is spent on modeling dimensionally reduced entities [2]. Automex [3] observed that the complexity involved in generating midsurfaces takes about 70–90 % of the pre-processing time. Thus, computing well-connected midsurface is highly critical to the industry.

Even if there is such a crucial need, the quality of midsurface generated by commercial CAD–CAE applications is not encouraging. Figure 3 shows the output of automatic midsurfacing by two of the leading CAD–CAE commercial applications. English alphabets are chosen as benchmarking examples, as they are easy to understand and represent a wide variety of shapes and connection types found in the real-life thin-walled parts.

Midsurface failures such as missing surfaces, gaps, not lying midway, etc. can be clearly seen in both the outputs (Fig. 3b, c). The present research work attempts to address these problems by proposing a new approach, as described in the following section.

4 Identification of problems and potential solutions

By far, the most widely and commercially used approach appears to be of midsurface abstraction (face pairing), but, as seen before, that also is not without any problems. Two of the most critical problems are as follows:

4.1 Face pairs detection problem

Face pairs, in a way, decompose the whole model into sub-volumes. Each sub-volume creates its own midsurface patch which, later, are "sewed" together. In the feature-based CAD model, at each feature step, a tool body is created and Booleaned to the existing model. Instead of

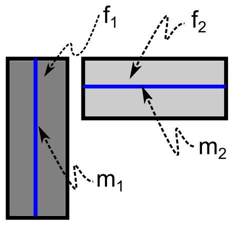


Fig. 4 Feature interaction

detecting face pairs in the final solid, feature information, at each step, can be leveraged to compute the midsurface patch for its corresponding sub-volume. Here, due to the large number of feature types, writing midsurface computation for all the feature types separately becomes a tedious task. This work solves this problem by generalizing these individual feature types into a generic form and then computes midsurface patches for just the generic type. Feature generalization called $\mathcal{ABLE}$ has been proposed to generalize the form features into a generic "loft" feature having profile(s) and a guide curve. Midsurface patch can then be computed by sweeping the midcurve of the profile(s) along the guide curve (Sect. 7.1).

4.2 Face–pair interaction problem

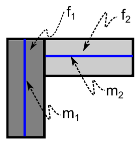
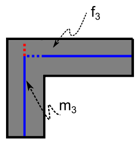
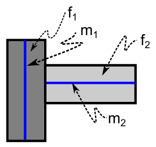
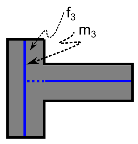
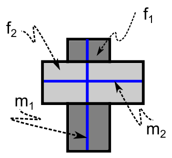
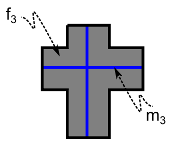
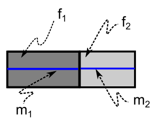
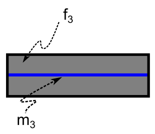
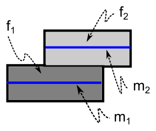
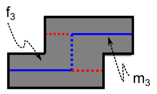
In traditional approaches, many midsurface patches fail to get joined, resulting in gaps, overlaps, etc. Midsurface patches need to be connected in the same manner as the face pairs in the input model are connected[1]. As there is no theoretical framework encompassing all possible interaction types, providing a unified logic has not been possible ([34]). Developing connection logic for each of these types separately can be a tedious task.

Figure 4 shows representative interaction among two sub-volumes along with their own midsurface patches. Sub-volumes f_1 and f_2 include m_1 and m_2 as their respective midsurface patches. These can interact in multiple ways, such as, 'L', 'T', 'X', Overlap, Align, etc. Table 1 shows how such interactions are dealt with, in the traditional approaches [9, 10, 28, 40] using extend trim so as to connect two midsurface patches.

It is desirable to provide a consistent generic method that can be applicable to a wide variety of the cases. Therefore, a feature-based cellular decomposition (FBCD) method is employed. It is a cellular decomposition in the context of features. Bidarra et al. [4–6] showed that feature-based

[1] Although diagrams used to explain are schematic with rectangles depicting feature volumes and lines depicting midsurfaces, in practice these can be of any general shape.

Table 1 Interaction of features and their respective midsurface patches

Type	Interaction	Midsurface	Joining rule
L			New midsurface needs to be extended to meet the existing midsurface. Extra patches of the existing midsurface are trimmed and removed
T			New midsurface needs to be extended to meet the existing midsurface. No trimming is needed
X			New midsurface does not need to be extended. No patch from the existing midsurface is removed
Align			No adjustment needed
Overlap			The extra patches are removed from both, the new as well as existing midsurface. New edges are joined with an additional patch

cellular model is a far richer representation than the usual feature-based CAD model. They employed it for representing multi-view product geometry, feature operations, etc. Chen et al. presented a unified feature modeling scheme using cellular topology [13]. Our research, while borrowing this notion of cells with owner features, goes a step further and assigns a generalized owner feature, thereby making it easier to write a generic algorithm. In addition, the FBCD used here, as compared to other approaches, is performed, not on the final solid, but in the context of the feature tree, incrementally at each feature step. Thus, the decomposition region remains local, minimizing generation of redundant cells. So, there is no need to utilize the maximal ([41]) volume strategies to merge a large number of cells computed during the global splittings.

Thin-walled parts are present in a variety of domains, such as plastics, sheet metal, machined components, etc. They are either constant thickness parts such as in sheet metal domain, or variable thickness parts, as in plastics domain. According to D.W. Brown Report [20], sheet metal parts constitute a large share, i.e approximately 40 % of the manufacturing processes. They are found in a wide variety of domains such as automobile, aircraft, shipbuilding, and consumer products. Thus, the present research work focuses on generating midsurface of sheet metal CAD models.

5 Overview of the proposed approach

With an objective to compute a well-connected midsurface, the input model undergoes thorough transformations as shown in Fig. 5 and Algorithm 1.

– *Input* In practice, CAD models exist in various representations such as mesh, solid, and feature-based CAD. As the present research leverages feature information, it expects a feature-based CAD model of a sheet metal part as the input. It is represented by ($\cup_q f^3$), where '$\cup$' denotes a collection of 'q' features ('f') having dimensionality '3' (solids). Insistence on the presence of features, at times, can be deemed as a limitation compared to far widely available solid model format called Brep (boundary representation). In such cases, techniques such as segmenta-

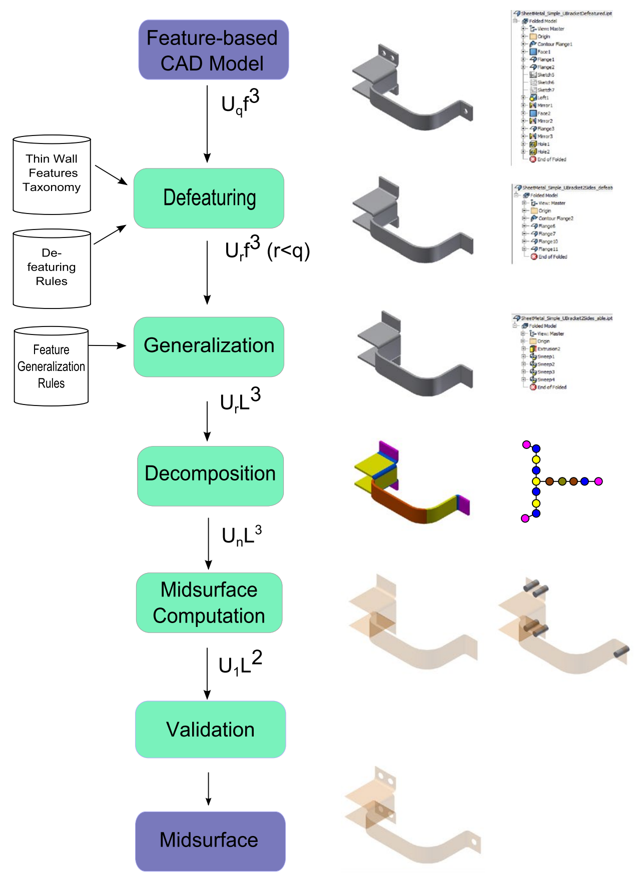


Fig. 5 The overall workflow

tion, cellular decomposition, and feature recognition (FR) can be used effectively to convert the Brep (non-feature-based) model to the feature-based CAD models.

- *Defeaturing* It has been observed that the gross shape yields better midsurface compared to the detailed model. Thus, the objective of this defeaturing module is to compute the gross shape by removing irrelevant features. Defeatured model is denoted as $(\cup_r f^3, r \leq q)$ (Sect. 5.1)
- *Generalization* Instead of working on a variety of features, it is effective to write feature-based algorithms on a generic feature form. This module proposes such transformation from sheet metal features to a generic loft equivalent feature. As mentioned in Sect. 4.1, such generalized model is known as $\mathcal{ABLE}$ and represented by $(\cup_r L^3)$ (more details in Sect. 5.2)
- *Decomposition* Cellular decomposition is performed on the generalized modelto form sub-voulmes, called "cells"having respective owner loft feature(s). A graph is populated with the cells at nodes(more details in Sect. 6).
- *Midsurface computation* The nodes are classified into midsurface patch-generating cells (called "solid cells" or *sCells*) and midsurface patch-joining cells(called "interface cells" or *iCells*). *sCells* generate midsurface patches and *iCells* connect them (more details in Sect. 7).
- *Validation* Midsurface needs to faithfully mimic the parent shape. A topological validation approach is proposed and used to assess the correctness of the midsurface.
- *Midsurface/output* A well-connected midsurface is then sent to downstream applications such as CAE analysis.

Algorithm 1 Feature-based midsurface computation

```
Require: Feature-based CAD model represented by $(\cup_q f^3)$
  $\cup_r f^3 = feature\_based\_defeaturing(\cup_q f^3)$, where, $r \leq q$
  $\cup_r L^3 = feature\_based\_generalization(\cup_r f^3)$
  $\cup_n C^3 = feature\_based\_cellular\_decomposition(\cup_r L^3)$, where, $C_i \cap C_j = 0|O^2_{i,j}$
  $G(n,) = compute\_graph\_nodes(\cup_n C^3)$
  $(Gn, e) = find\_overlaps\_generate\_edges(G(n,))$
  $(sCell, iCell) = categorize\_cells((Gn, e))$
  for all $sCell$ do
    $\cup_v L^2 = compute\_midsurface\_patch(sCell)$ (Algorithm 2)
  end for
  for all $iCell$ do
    $\cup_w L^2 = resolve\_interactions(iCell)$ (Algorithm 3)
  end for
  return $\cup_1 L^2 = (\cup_v L^2) \cup (\cup_w L^2)$
```

The following sections provide details of the above-mentioned modules.

5.1 Defeaturing

Defeaturing means removal of irrelevant features ([36]). The selection of the irrelevant features is decided by three proposed phases. In Phase I, sheet metal domain-specific features are removed based on newly proposed taxonomy. In Phase II, the relative size of the remnant feature volumes is used to decide the eligibility for removal. In Phase III, a novel idea of caching bodies of relevant negative features is used. Features, like holes and cuts if by usual rules do not get removed, they are still removed and their respective tool bodies are preserved. These bodies, called "dormant bodies", are then used in the last stage to pierce the final midsurface. With this arrangement, computing midsurface patches is simplified due to lack of holes, etc., while re-piercing of the cached dormant bodies ensures their faithful representation back in the midsurface.

5.2 Generalization

Features not only carry shape information such as geometry and topology, but also embed meta-information based on the application context ([30]). With a variety of applications, each having its own need has resulted in various feature schemes not only in different CAD applications, but also in various environments present within the same CAD application. This causes a plethora of cases to deal with while designing feature-based algorithms. Generalization alleviates this problem by extracting a generic feature form. In the context of current research, such generic CAD model is known as $\mathcal{ABLE}$(affine transformations, Booleans, lofts and entities). In this module,,input sheet metal features are converted into a loft equivalent features in the $\mathcal{ABLE}$ model.[2]

6 Decomposition

As mentioned in Sect. 5, the proposed approach starts with a feature-based CAD model. The model is defeatured by applying a set of strategies so that the output CAD model has small or irrelevant features removed. The feature tree of defeatured CAD model is transformed to loft-feature-tree. The advantage is that the computation logic of midsurface patches needs to be based on only the "loft" feature, that is, profile(s) and a guide curve (Algorithm 2). But in such a part, each loft feature is not a separate entity, but is Booleaned at each tree node to the existing model till that state. Detecting and separating the common portions will be needed to decide on the midsurface-generating sub-volumes and patch-joining sub-volumes. This splitting is done by decomposition.

The present research work uses cellular decomposition in the context of features and is thus called feature-based cellular decomposition (FBCD) with primary rules as:

- *Feature partitioning* Internal as well as external Booleans are changed to the "New Body" type, so that the tool body volumes gets separated. The volumes may still overlap spatially with each other.
- *Concave edge partitioning (CEP)* Overlapping volumes are split at concave edges. Faces at these edges are extended and used as partitions to split. Research in cellular decomposition, especially for computer-aided manufacturing (CAM) and CAE has been going on for decades. Feature-based cellular decomposition, which deals with either decomposition of features or feature recognition after decomposition, has also been researched extensively [4, 6, 10, 24, 37, 38, 40, 42]. There have been a few attempts to compute a midsurface using cellular decomposition as well [9, 14, 39, 43] The present research leverages past CEP works with an enhancement that the face extensions are not done infinitely (or covering the part's bounding box), but within the influence zone decided by two interacting features.

Output cells do not overlap volumetrically, but can touch others at the boundary faces, only fully, but not partially. Figure 6a shows the input model and Fig. 6b shows the output of FBCD, of the CAD model of a real-life sheet metal part. Figure 6c shows the schematic figure of an "L"-shaped model, where two features f_1 and f_2 are interacting. After feature partitioning, f_1 and f_2 get separated, though they lie spatially in the same locations, thus overlapping volumetrically. CEP happens at the concave edges. In the above scenario, volumes of f_1 and f_2 are partitioned in such a way that a common cell (with owner $f_1''f_2''$) is formed (Fig. 6d). The remaining portion of f_1 and f_2 are termed as f_1' and f_2', respectively. Overlapping faces are denoted as O_1 and O_2, where O denotes "overlap" and $_1$ and $_2$ are the instances. Thus, now, the cell bodies do not overlap volumetrically $[O^3_{i,j} = 0]$ (meaning, overlap between bodies $_i$ and $_j$ which is of dimensions ^{3}, that is, volumetric), but may overlap at faces $[O^2_{i,j}]$ (meaning, overlap between bodies $_i$ and $_j$, which is of dimensions ^{2}, that is, surface wise) fully and not partially, denoted as $[\ C_i \cap C_j = 0 | O^2_{i,j}]$ (meaning, intersection $\cap$ of cells C_i and C_j is either nothing or full surface, but not partial).

FBCD results in n cells, each assigned with owner loft feature(s). A cell adjacency graph $[CAG = G(n, e)]$ is formed with nodes representing cells. Each face overlap between two nodes is represented by an edge (e). Figure 7 shows the CAG of simple cell bodies configuration of an "L"-shaped part. Node n_1 corresponds to a cell body, with owning feature f_1', whereas node n_3 corresponds to the cell body with owning feature f_2'. The common cell body owned by $f_1''f_2''$ is represented by node n_2. Edge e_{12} corresponds to the overlapping face O_1, whereas Edge e_{23} corresponds to the overlapping face O_2.

The advantage of the CAG representation is that it is easier to classify nodes and delegate specific computational work to them. Looking at the connectivity at each cell

[2] Extrude, Revolve and Sweep are considered as variations of loft. Sweep has a single profile and a singe guide curve, whereas loft can have multiple profiles and a guide curve. In the context of current research, which is focused on sheet metal parts having constant thickness, Sweep with a single profile and a guide curve is sufficient to represent most of the features.

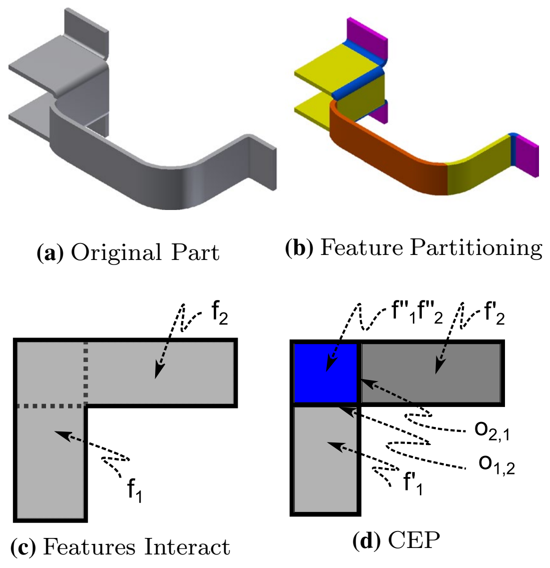


Fig. 6 Feature-based cellular decomposition

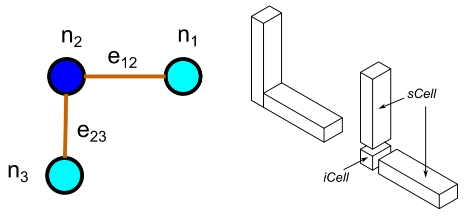


Fig. 7 Feature-based cellular graph

nodes, they are classified into solid cells (*sCell*) and interface cells (*iCell*). Generating midsurface patches is delegated to the *sCell*s and the problem of resolving interaction among the midsurface patches is carried out by the *iCell*s. The classification strategy, based on the simple graph topology rules, simplifies the complexity of the midsurface generation problem to a great extent. Thus, the present research work does not need to enumerate specific heuristic rules used for specific connection types, or underlying geometry types. The following section elaborates more on this proposed approach of computing midsurface

7 Midsurface computation

Each node in CAG represents a cell which has three dimensions, two sides aligned to the bounding box of the profile and the third along length of the guide curve (say, d_1, d_2, d_3).

Nodes are classified as *sCell*s and *iCell*s as per the following definitions:

Definition 1 Solid cell (*sCell*) is a cell body with only one of its dimensions (d_1) less than the threshold factor (t) times any of the other two dimensions (d_2, d_3), denoted as $d_1 < t.d_2 \quad \& \quad d_1 < t.d_3$.

Definition 2 Interface cell (*iCell*) is a cell body with both minimum two dimensions (d_1, d_2) less than the threshold factor (t) times of the maximum dimensions (d_3), denoted as $d_1 < t.d_3 \quad \& \quad d_2 < t.d_3$.

In Fig. 7, n_1 and n_3 are solid cells (*sCell*), whereas n_2 is an interaction cell (*iCell*). The *sCell* computes the midsurface patches, whereas the *iCell* connects all the midsurface patches incident on them. Both processes are detailed out in the sections below.

7.1 Computing midsurface patches in *sCell*s

Algorithm 2 describes the procedure of computing the midsurface patches in the *sCell*s.

The patch is generated based on the relative sizes of sketch profile p and the guide curve g of the owner loft feature of the *sCell*. Depending on the aspect ratio or the "thinness" criteria of the *sCell*, there are three possibilities, viz., long guide, short guide and equal guide. For example, 'thinness' criterion used by Woo [39] is $\frac{\min(L,H)}{d} > X, 1.2 \leq X \leq 3$, where L is the length, H is the height and d is the thickness of the shape. The present research work uses threshold as 2.

Figure 8a shows an Extrude feature (a loft equivalent) having a profile and a guide curve where the guide is far shorter than the size of the profile. This case is called as "short guide". Because profile is a 2D entity, its size comparison with 1D guide curve cannot be done directly. So, the perimeter of the profile with some scaling factor is compared with the length of the guide. This size comparison determines the 'thinness' of the *sCell*.

Figure 8b shows a similar Extrude, but which is built by extruding a smaller profile along a longer guide direction. This case is called "long guide".

Figure 8c shows an Extrude whose profile and guide sizes are similar. This case is called "equal guide" and the cell is "thick cell".

Based on the above-mentioned type of the *sCell*, different midsurface patch generation techniques are used, as described below:

– **Long guide** *sCell* : *Midcurve* is extracted from the *profile* and swept along the *guide* to generate the midsur-

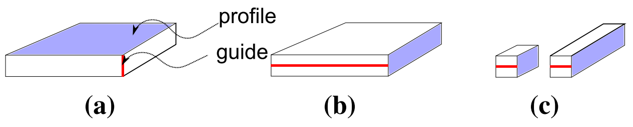


Fig. 8 Types of *sCell*

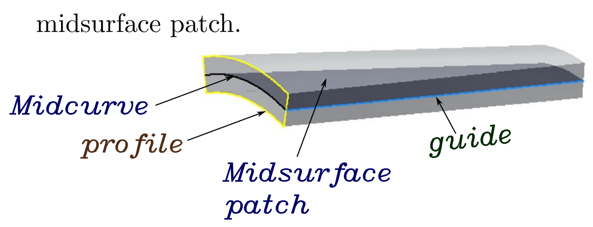


Fig. 9 Midsurface patch creation in the "long guide" case

face patch. Figure 9 shows a solid cell having a profile, a guide, a midcurve computed and the midsurface patch generated.

- **Short guide** *sCell* : The *profile* face itself is *offset* along half of the distance of the *guide*. *Midcurve* is not needed to be computed in this case.
- **Equal guide** *sCell* : This is a thick cell and thus the midsurface patch is not generated for it. During midsurface patch-joining phase, such thick cells may participate as patch-joining cells.

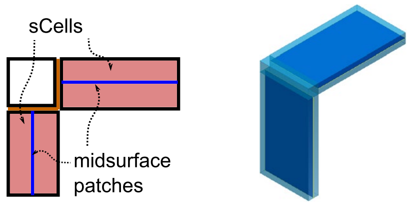
Applying these strategies for *sCell*s, the corresponding midsurface patches generated are as shown in Fig. 10. Once all the *sCell*s are done with the computation of midsurface patches, the next step is to resolve interactions between these patches in *iCell*s.

Algorithm 2 sCell midsurface patch computation

```
Require: $sCell$
1: $f = sCell \rightarrow owning\_feature()$
2: $p = f \rightarrow get\_profile()$
3: $g = f \rightarrow get\_guide()$
4: if $is\_long\_guide(p, g) == true$ then
5:     $m^1 = p \rightarrow compute\_midcurve()$
6:     $m^2 = sweep(m^1, g)$
7: else if $is\_short\_guide(p, g) == true$ then
8:     $m^2 = offset(p, g/2)$
9: end if
```

Algorithm 2 describes the process in the pseudo-code form. Lines 1–3 show the extraction of the owner feature of *sCell*, of profile and guide from the owner feature. Line 4 condition tests if the cell is of long guide type and generates midcurve m^1, where 1 stands for dimension 1, representing a curve. m^1 is swept along guide g to generate the midsurface patch m^2, where 2 stands for dimension 2, representing a surface. Line 7 condition tests if the cell is of short guide type and then generates a midsurface patch m^2 by offsetting profile face p, by half the guide g.

At this stage, all the *sCell*s are filled with the midsurface patches and are ready to get connected in *iCell*s as described in the following section.



(a) Patches, 2d view **(b)** Patches, 3d view

Fig. 10 Midsurface patches

7.2 Resolving interactions between midsurface patches in *iCells*

Input to this module is a CAG with midsurface patches computed at all the *sCell*s. The sole responsibility of *iCell*s is to connect the midsurface patches incident on them, either by extending the midsurface patches from the adjacent *sCell*s or by generating new ones. The CAG is used to traverse nodes of *iCell*s one by one. For each *iCell*, the midsurface patch interactions are resolved as follows (Algorithm 3)[3]:

Each *iCell* is connected with adjacent cells via edges e(s). Each incident e has two nodes (like e_{12} has n_1 and n_2 in Fig. 7), where one is the "self", the current *iCell*, and the second one is called the "adjacent node". The "adjacent node" could be either a *sCell* or an *iCell*.

If the "adjacent node" is a *sCell*, the adjacent midsurface patch is extended up to *iCell*'s centroid (Fig. 11b). An intersection curve (m^1) is computed between the overlapping face ($O_{1,2}$) and the midsurface (m^2). This curve acts as midcurve for the extension into the *iCell*. In case where all the three cells are geometrically flat and relatively big, then all are marked as *sCell*s (not *iCell*s) and have no extensions.

If the "adjacent node" is an *iCell*(n_3), an extra patch is created between the centroids of the two *iCell*s (Fig. 11c). Thus, all the *iCell*s join the midsurface patches (Fig. 11a) to form a well-connected midsurface.

[3] Although the diagrams used to explain are schematic, showing rectangular shapes with just two incident edges, the algorithm presented is applicable to a variety of shapes and more number of incident edges, as shown in Table 2.

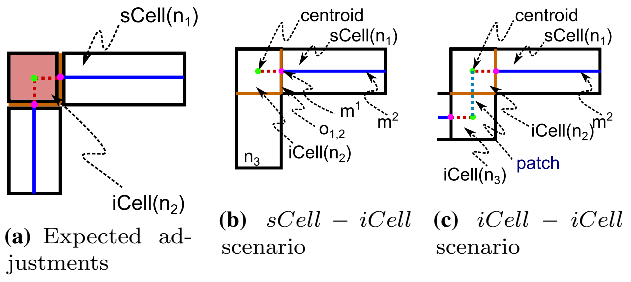


Fig. 11 Resolving interactions in the *iCell*

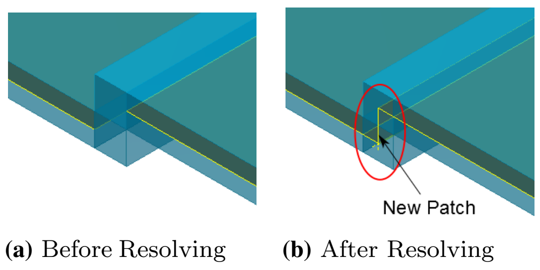


Fig. 12 *iCell* resolving in overlap case

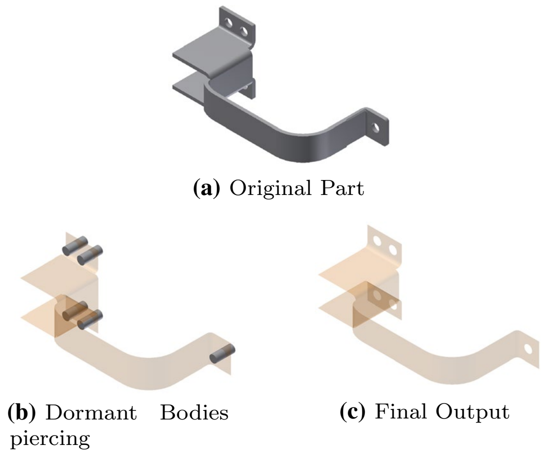


Fig. 13 Reapplication of dormant feature tool bodies

Figure 11b shows the working of *sCell–iCell* interaction. Midsurface patch from a $sCell(n_1)$ needs to be extended to a common location, say, centroid of the $iCell(n_2)$, where n_1, n_2 denotes the corresponding cellular graph nodes.

Figure 11c shows the working of *iCell–iCell* interaction. Midsurface patch from a $sCell(n_1)$ needs to be extended to a common location, say, centroid of the $iCell(n_2)$. Similar extension has to be computed from the *iCells* pairing with the other *sCell*. Centroids of both the *iCells* need to be connected with a new patch.

Figure 12 shows the *iCell–iCell* interaction in 3D. Figure 12a shows the state before joining of the patches $m1$, $m2$ whereas Fig. 12b shows the state of the model with connected midsurface patches.

Algorithm 3 *iCell* midsurface patch interaction resolution

```
Require: $iCell$
1: while $size(iCell \rightarrow edges) > 0$ do
2:    $e_i = iCell \rightarrow edge$
3:    $n_s = iCell$
4:    $n_o = e_i \rightarrow get\_adjacent\_node(n_s)$
5:    if $n_o \rightarrow type == sCell$ then
6:       $m_o = n_o \rightarrow query\_midsurface()$
7:       $O_f = e_i \rightarrow overlapping\_face()$
8:       $m^1 = surf\_surf\_intersection(O_f, m_o)$
9:       $m^2 = extrude(m^1, n_o \rightarrow centroid)$
10:   else if $n_o \rightarrow type == iCell$ then
11:      $c_1^1 = get\_curve\_at\_centroid(n_s)$
12:      $c_2^1 = get\_curve\_at\_centroid(n_o)$
13:      $m^2 = create\_patch(c_1^1, c_2^1)$
14:   end if
15: end while
```

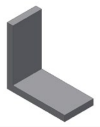
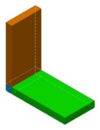
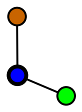
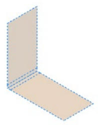
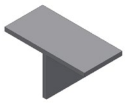
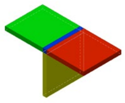
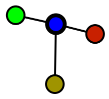
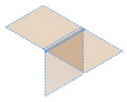
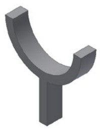
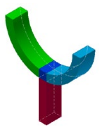
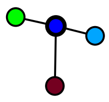
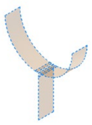
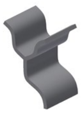
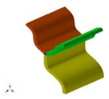
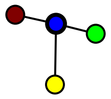
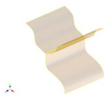
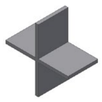
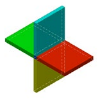
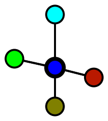
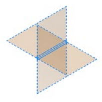
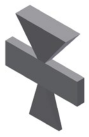
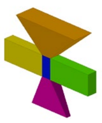
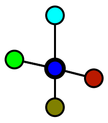
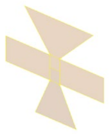
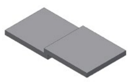
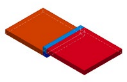
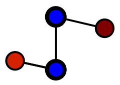
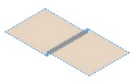
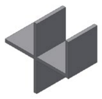
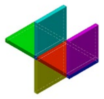
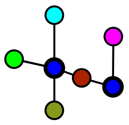
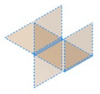
Algorithm 3 starts with the input as an *iCell*. Edges incident on the *iCell* are iterated (Algorithm 3 line: 1/15 loop). The current edge (e_i) and the node corresponding to *iCell* (called n_s, where $_s$ is for "self") are extracted. The other node pointed by e is called n_o, where $_o$ stands for "other". Condition block in lines 5–9 is for *sCell–iCell* interaction, whereas lines 10–13 is for *iCell* – *iCell* interaction. Extensions can either be done by extending existing midsurface patch or by creating a new patch by extrude. The second choice is used, as shown in lines 6–9. Lines 11–13 show creation of new patch joining existing extensions from both the patches up to centroids.

Once all interactions are resolved in *iCells*, the result is a well-connected midsurface. Cached dormant bodies are pierced into the midsurface (Sect. 5.1) to restore the temporarily removed negative features (Fig. 13).

Table 2 shows various cases, with their respective cellular representations, graphs and midsurface outputs.

The present research work has been implemented based on Autodesk Inventor Application Programming Interfaces (APIs), which introduces certain limitations in terms of modeling functionalities. Apart from this, the proposed approach leverages already existing techniques such as convex partitioning, which brings its own limitations. The quality of midsurface output depends on the appropriateness of cellular decomposition and feature assignments, and it may not be possible to represent a cell as loft equivalent feature. In such cases, face pair logic and interpolation of the faces in the cell, is leveraged to generate the midsurface patches.

Table 2 Feature-based cellular midsurface

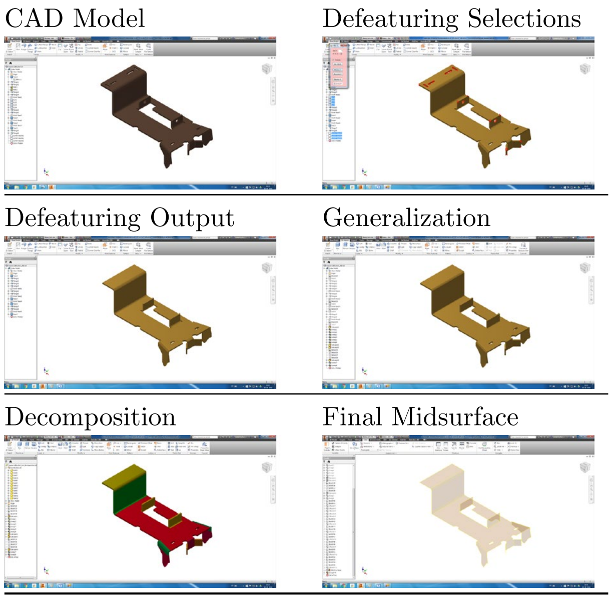
Part	Cellular	Graph	Result
			
			
			
			
			
			
			
			

Barring these limitations, the proposed approach works effectively as demonstrated by real-life cases in the following section.

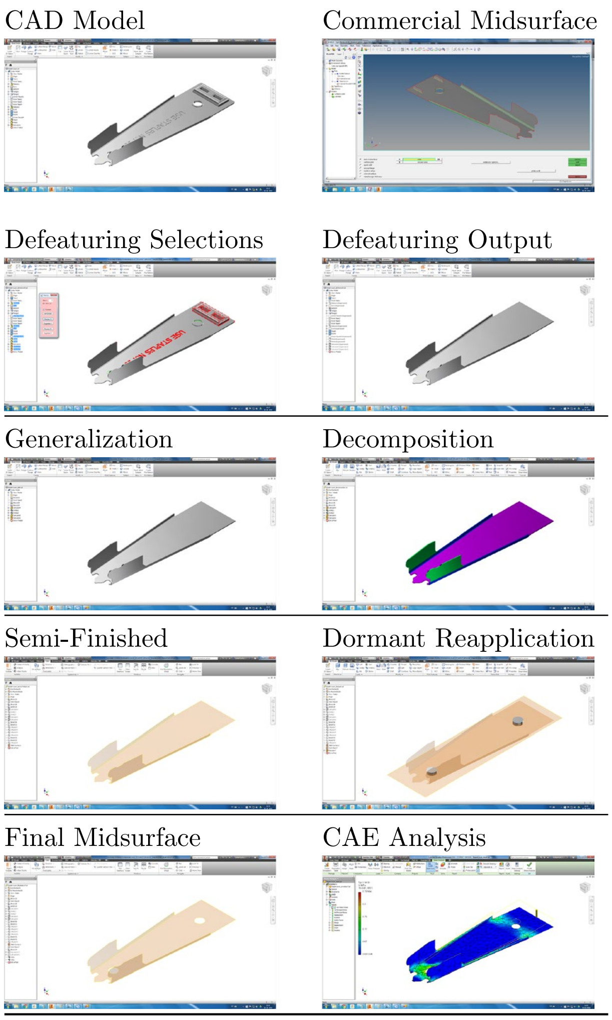
8 Results

The proposed approach presented here has been implemented using the Autodesk® InventorAPIs in Visual Basic. Net on an Intel i3 64-bit 2 GHz Windows 7 machine. Following are some of the real-life test cases demonstrating the efficacy of the proposed approach.

8.1 The standing bracket



8.2 The stapler bottom



The superiority of the proposed approach is in the manner in which it has abstracted both the sub-problems, that is, computation of midsurface patches and resolving interactions among them. Hard-coded rules based on specific surface types or connection configurations are avoided, making the whole process generic and adaptable in a wide variety of configurations.

9 Conclusion

Midsurface is one of the widely used idealizations for CAE analysis of thin-walled models. A study of the existing method revealed critical insights into the causes of midsurface errors such as gaps, overlaps, and not lying midway. These errors, in one of the most widely used method, called midsurface abstraction (MA), are due to issues in face pair detection which result in problems of midsurface patch generation and midsurface patch joining. For midsurface patch generation problems, the present research proposes feature generalization approach known as $\mathcal{ABLE}$. For issues in joining midsurface patches, the present research proposes the use of feature-based cellular decomposition in developing a generic logic, without limiting to any specific types of connections.

Overall, the proposed system simplifies the input sheet metal feature-based CAD model to its gross shape. It transforms the remaining features to generic loft features. The generalized $\mathcal{ABLE}$ is then decomposed into cells with owner loft feature(s). Feature parameters are leveraged to compute midsurface patches and a generic joining logic is used to connect the patches. The proposed methodology scores over existing approaches [8–12, 14, 39, 43] in terms of simplifying the complexities to a great extent and solving them rapidly. It is not restricted by the underlying geometries as well as a few hard-coded connection types.

The proposed approach was tested extensively on a wide range of academic as well as industrial sheet metal parts and it was observed that the midsurfaces were generated. There were a few pathological cases which the algorithms were not able to handle. However, such corner cases were very limited and further advancements in the algorithms(not scoped in this work) can resolve these as well.

References

1. Attali D, Montanvert A (1997) Computing and simplifying 2D and 3D continuous skeletons. Comput Vis Image Underst 67:261–273. doi:10.1006/cviu.1997.0536
2. Austreng R (2007) Rapid generation of finite element models from 3d cad. Tech. rep, Altair Hyperworks
3. Backes A, Glockner C, Schwanecke U (2014) Automatic midsurface generation as a basis for technical simulation. ATZ Worldwide 116(2):20–23. doi:10.1007/s38311-014-0019-0
4. Bidarra R, Teixeira JC (1993) Intelligent form feature interaction management in a cellular modeling scheme. In: Symposium on solid modeling and applications, pp 483–485. doi:10.1145/164360.164543
5. Bidarra R, Dohmen M, Bronsvoort WF (1997) Automatic detection of interactions in feature models. In: CD-ROM proceedings of the 1997 ASME design engineering technical conferences, pp 14–17
6. Bidarra R, Kraker KJD, Bronsvoort WF (1998) Representation and management of feature information in a cellular model. Comput Aided Des 30:301–313. doi:10.1016/S0010-4485(97)00070-5
7. Blum H (1967) A transformation for extracting new descriptors of shape. In: Wathen-Dunn W (ed) Proc. models for the perception of speech and visual form. MIT Press, Cambridge, MA, pp 362–380. http://pageperso.lif.univ-mrs.fr/~edouard.thiel/rech/1967-blum.pdf
8. Boussuge F (2006) Idealization of cad assemblies for fe structural analyses. PhD thesis, University of Grenoble
9. Boussuge F, Léon JC, Hahmann S, Fine L (2014a) Extraction of generative processes from b-rep shapes and application to idealization transformations. Comput Aided Des 46:79–89. doi:10.1016/j.cad.2013.08.020
10. Boussuge F, Lon JC, Hahmann S, Fine L (2014b) Idealized models for fea derived from generative modeling processes based on extrusion primitives. In: Proceedings of the 22nd international meshing roundtable pp 127–145. doi:10.1007/978-3-319-02335-9-8
11. Cao W, Wu H, Jiang Y, Liu Y, Gao S (2009) Representation and automated generation of analysis feature model for finite element analysis. In: Proceedings of the ASME 2009 international design engineering technical conferences & computers and information in engineering conference vol 5, pp 605–614. doi:10.1115/DETC2009-86322
12. Cao W, Chen X, Gao S (2011) An approach to automated conversion from design feature model to ananlysis feature model. In: Proceedings of the ASME 2011 international design engineering technical conferences & computers and information in engineering conference vol 5, pp 655–665. doi:10.1115/DETC2011-47555
13. Chen G, Ma YS, Thimm G, Tang SH (2006) Using cellular topology in a unified feature modeling scheme. Comput Aided Des Appl 3:89–98
14. Chong CS, Kumar AS, Lee KH (2004) Automatic solid decomposition and reduction for non-manifold geometric model generation. Comput Aided Des 36:1357–1369. doi:10.1016/j.cad.2004.02.005
15. Dabke P, Prabhakar V, Sheppard S (1994) Using features to support finite element idealizations. Comput Eng 1:183–183
16. Donaghy R, Cune W, Bridgett S, Armstrong C, Robinson D, McKeag R, Robinson D, et al (1996) Dimensional reduction of analysis models. in: Proceedings 5th international meshing roundtable pp 307–320
17. Donaghy R, Armstrong C, Price M (2000) Dimensional reduction of surface models for analysis. Eng Comput 16(1):24–35. doi:10.1007/s003660050034. http://www.springerlink.com/link.asp?id=u2r7wpdrckc60f2v
18. Fischer A, Wang KK (1997) A method for extracting and thickening a mid-surface of a 3D thin object represented in NURBS. J Manuf Sci Eng Trans ASME 119:706–712. doi:10.1115/1.2836813
19. Fischer A, Smolin A, Elber G (1999) Mid-surface of profile-based freeforms for mold design. J Manuf Sci Eng Trans ASME 121:202–207. doi:10.1115/1.2831206

20. Halpern M (1997) Industrial requirements and practices in finite element meshing: a survey of trends. In: 6th International meshing roundtable, pp 399–411
21. Hamdi M, Aifaoui N, Benamara A (2006) Design and analysis integration model based on idealization of cad geometry. In: The international conference on advances in mechanical engineering and mechanics (ICAMEM)
22. Lam L, Lee SW, Suen CY (1992) Thinning methodologies-a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 14(9):869–885. doi:10.1109/34.161346
23. Lee H, Nam YY, Park SW (2007) Graph-based midsurface extraction for finite element analysis. In: Computer supported cooperative work in design, 2007. CSCWD 2007. 11th international conference on, IEEE, pp 1055–1058
24. Lee JY, Lee JH, Kim H, Hyung-Sun K (2004) A cellular topology-based approach to generating progressive solid models from feature-centric models. Comput Aided Des 36:217–229
25. Li M, Zheng J, Martin RR (2012) Quantitative control of idealized analysis models of thin designs. Comput. Struct. 106:144–153
26. Lockett H, Guenov M (2008) Similarity measures for midsurface quality evaluation. Comput Aided Des 40(3):368–380. doi:10.1016/j.cad.2007.11.008
27. Ramanathan M, Gurumoorthy B (2004) Generating the mid-surface of a solid using 2d mat of its faces. Comput Aided Des Appl 1:665–674. doi:10.1080/16864360.2004.10738312
28. Rezayat M (1996) Midsurface abstraction from 3D solid models: general theory, applications. Comput Aided Des 28:905–915. doi:10.1016/0010-4485(96)00018-8
29. Robinson TT, Armstrong CG, McSparron G, Quenardel A, Ou H, McKeag R (2006) 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, ACM, pp 117–128, doi:10.1145/1128888.1128905
30. Shah JJ, Mäntylä M (1995) Parametric and feature-based CAD/CAM: concepts, techniques, and applications, 1st edn. Wiley, New York. http://dl.acm.org/citation.cfm?id=546488
31. Sheen DP, Son Tg, Ryu C, Lee SH, Lee K (2009) Dimension reduction of solid models by mid-surface generation. Int J CAD/CAM 7(1):71–80. http://ijcc.org/ojs/index.php/ijcc/article/view/79/72
32. Sheen DP, Son Tg, Myung DK, Ryu C, Lee SH, Lee K, Yeo TJ (2010) Transformation of a thin-walled solid model into a surface model via solid deflation. Comput Aided Des 42(8):720–730. doi:10.1016/j.cad.2010.01.003
33. Steele CR, Balch CD (2009) Stretching and bending of plates—fundamentals. http://web.stanford.edu/ chasst/Course%20Notes/Introduction%20to%20the%20Theory%20of%20Plates.pdf
34. Stolt R (2006a) Reusing cad models for die-casting products for fea. In: Proceedings of 2nd NAFEMS seminar: prediction and modelling of failure using FEA
35. Stolt R (2006b) A sectioning method for constructing the mid-surface of thin walled die-cast and injection moulded parts. Tech. rep., Jonkoping University. http://www.diva-portal.org/smash/get/diva2:4318/FULLTEXT01.pdf
36. Thakur A, Banerjee AG, Gupta SK (2009) A survey of CAD model simplification techniques for physics-based simulation applications. Comput Aided Des 41:65–80. doi:10.1016/j.cad.2008.11.009
37. Van Treeck C, Romberg R, Rank E (2003) Simulation based on the product model standard ifc. In: Proc. 8th int. IBPSA conf. building simulation, Eindhoven, Netherlands, vol 142
38. Woo Y (2003) Fast cell-based decomposition and applications to solid modeling. Comput Aided Des 35:969–977. doi:10.1016/S0010-4485(02)00144-6
39. Woo Y (2014) Abstraction of mid-surfaces from solid models of thin-walled parts: a divide-and-conquer approach. Comput Aided Des 47:1–11. doi:10.1016/j.cad.2013.08.010
40. Woo Y, Kim SH (2014) Protrusion recognition from solid model using orthogonal bounding factor. J Mech Sci Technol 28(5):1759–1764. doi:10.1007/s12206-014-0322-0
41. Woo Y, Sakurai H (2002) Recognition of maximal features by volume decomposition. Comput Aided Des 34:195–207. doi:10.1016/S0010-4485(01)00080-X
42. Wu H, Shuming G (2014) Automatic swept volume decomposition based on sweep directions extraction for hexahedral meshing. In: 23rd international meshing roundtable vol 82, pp 136–148. doi:10.1016/j.proeng.2014.10.379
43. Zhu H, Shao Y, Liu Y, Li C (2015) Mid-surface abstraction for complex thin-wall models based on virtual decomposition. Int J Comput Integr Manuf 29(8):821–838. doi:10.1080/0951192X.2015.1068455