

## Computation of Midsurface by Feature-based Simplification-Abstraction-Decomposition

Yogesh H Kulkarni \*

Anil Sahasrabudhe

PhD Student  
College of Engineering Pune, India  
Email: kulkarniyh12.mech@coep.ac.in

Director  
College of Engineering Pune, India  
Email: anil.sahasrabudhe@gmail.com

Mukund Kale

Manager, Siemens PLM, India. Email: mukund\_kale@hotmail.com

Computer-aided Design (CAD) models of thin-walled solids such as sheet metal or plastic parts are often reduced dimensionally to their corresponding midsurfaces for quicker and fairly accurate results of Computer-aided Engineering (CAE) analysis. Computation of the midsurface is still a time-consuming and mostly, a manual task due to lack of robust and automated techniques. Most of the existing techniques work on the final shape (typically in the form of Boundary representation, B-rep). Complex B-reps make it hard to detect sub-shapes for which the midsurface patches are computed and joined, forcing usage of hard-coded heuristic rules, developed on a case-by-case basis. Midsurface failures manifest in the form of gaps, overlaps, non-mimicking input model, etc., which can take hours or even days to correct. The research presented here proposes to address these problems by leveraging feature-information available in the modern CAD models, and by effectively using techniques like simplification, abstraction and decomposition.

In the proposed approach, first, the irrelevant features are identified and removed from the input feature-based CAD model to compute its simplified gross shape. Remaining features, then undergo abstraction to transform into their corresponding generic Loft-equivalents, each having a profile and a guide curve. The model is then decomposed into cellular bodies and a graph is populated, with cellular bodies at the nodes and fully-overlapping-surface-interfaces at the edges. The nodes are classified into midsurface-patch generating nodes (called 'solid cells' or *sCells*) and interaction-resolving nodes ('interface cells' or *iCells*). In a *sCell*, a midsurface patch is generated either by offset or by sweep-

ing the midcurve of the owner-Loft-feature's profile along with its guide curve. Midsurface patches are then connected in the *iCells* in a generic manner, thus resulting in a well-connected midsurface with minimum failures. Output midsurface is then validated topologically for correctness. At the end of the paper, real-life parts are used to demonstrate the efficacy of the proposed approach.

**Keywords:** Midsurface, CAD, CAE, Cellular Decomposition, Model Simplification, Sheet Metal Features, Topological Validation, Feature Abstraction

### 1 Introduction

Getting a quicker validation of the proposed product is crucial in the era of fierce competition and faster obsolescence. Digital product development, which includes, modeling by CAD and analysis by CAE plays a critical role in quicker "Time to market". For sheet-metal, plastic products (generically classified as 'thin-walled'), a quicker and fairly accurate CAE analysis is possible by idealizing their solid models to the equivalent surface representation called "Midsurface". Such idealization is extremely important as stated by a Sandia report, which says that for complex engineering models, idealized models creation amounts to about 60% of overall analysis time, whereas the mesh generation consumes about 20% and solving the actual problem takes about 12% [1].

Midsurface can be envisaged as a surface lying midway of a thin-walled solid, and mimicking its shape. In CAE analysis, instead of using expensive 3D elements, 2D elements are used on the midsurface for fairly accurate results in far lesser computations and time. Because of this advantage, midsurface is widely used and is available in many commer-

\*Address all correspondence related to ASME style format and figures to this author.

cial CAD/CAE packages. Even in this age of scalable and near-infinite computing power, it is still desirable to idealize so as to run more design iterations quickly.

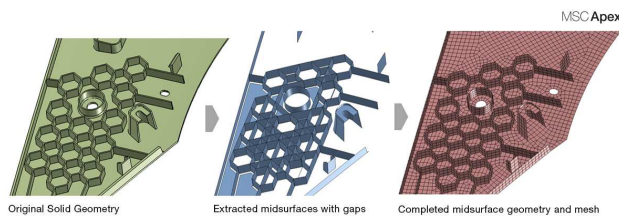


Figure 1: Midsurface with Errors (Source: [2]. Image, Copyright, MSC Software Corporation. Used with Permission.)

In spite of its demand and popularity, the existing techniques fail to compute a well-connected midsurface, especially for non-trivial shapes [3–5]. Failures manifest in the form of gaps, missing patches, overlapping surfaces, not lying midway, not mimicking the input shape, etc. (Figure 1). Correcting these errors is mostly a manual, tedious and highly time-consuming task, requiring hours to days. This correction time can be nearly equivalent to the time it can take to create the midsurface manually from scratch [4].

One of the major impediments in the development of the algorithm for automatic computation of the midsurface, is the lack of its precise definition [6]. Expectations vary based on the application context (Figure 2). For applications such as CAE, Figure 2b shows required midsurface, where there are no sharp corners, else there will be unnecessary dense meshing, where as Figure 2c is best suited for shape matching/retrieval. Figure 2d may be used when a disconnect needs to be highlighted. The variations shown in Figure 2 pertain to the geometries computed at the interfaces, and the proposed method can cater to them too, with additional rules (for future scope).

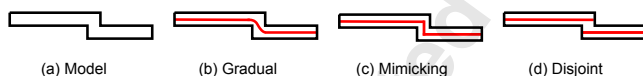


Figure 2: Expectations about Midsurfaces

The motivation of this work is to address midsurface problems by leveraging feature-based simplification, abstraction and decomposition. Although feature-based CAD modeling paradigm is prevalent for decades, many of the processes, like defeaturing, midsurface computation, etc. do not leverage it and are based on the final B-rep. Reason being, typically when CAD models are transferred to CAE, feature information is either removed (for proprietary reasons), or neutral formats without feature supports are used. With far more integrated CAD-CAE environments, feature-information access is getting available through Application

Programming Interfaces (APIs) and thus, feature-based algorithms can leverage them, as demonstrated in this work.

Many of the earlier works for computing midsurface were limited in the geometries they could support (say, only planar or analytic surfaces), features they could support (say, only extruded shapes, positive primitives), connection types they could handle (say, only parallel, or perpendicular), etc. By proposing a ‘generic’ methodology here, it is implied that, in principle, no such restrictions are present in this approach. It can be made to cater to any geometry types, feature types or connection types. Advantages can be seen in the range of shapes handled as well as in the minimization of failures (Table 20).

## 2 Related Work

For computing the midsurface, this work uses feature-based techniques in the domains of simplification/defeaturing, generalization/abstraction, cellular decomposition and topological validation. Following sub-sections evaluate some of the relevant past works in these domains and try to extract specific issues for addressal in this research.

### 2.1 Midsurface

Midsurface is a special case of a more generic structure called “Medial Object”. Medial Object is the geometry that is mid-way inside an input shape. The input shape can be an image, a 2D profile or a 3D solid. One of the early works in the field of Medial object was by a theoretical biologist, Harry Blum in 1967 [7]. Some of the most researched techniques are Medial Axis Transform (MAT), Chordal Axis Transform (CAT), Thinning, Pairing (Medial Abstraction, MA), etc. (Figure 3). Two of the most popular techniques amongst them are MAT and MA.

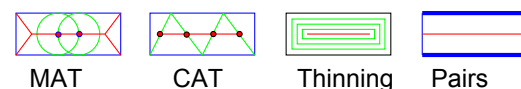


Figure 3: Medial Computation Techniques

MAT is a locus of the center of an inscribed maximal disc as it rolls around the object’s interior. As a formal definition is available, it can be generated for any shape, but the major drawback of this technique is that it creates unnecessary branches and gives unpredictable output due to perturbations on the input shape.

MA, initiated by Rezayat [8], involves identifying “face pairs”, constructing midsurface patches for each and then building a connected midsurface by extending or trimming and then joining the patches. This surface-pairing approach has benefits over MAT techniques because the resultant geometry is cleaner and requires lesser post-processing [5]. It has difficulty in identification of the surface pairs as well as in joining the midsurface patches. Many of the commercial

CAD-CAE packages use this approach and thus, have to provide manual/semi-automatic methods to correct the failures.

## 2.2 Defeaturing/Simplification

Defeaturing indicates removal of unwanted features. Thakur et al. [9] surveyed and classified various techniques used till then, into categories such as surface-based, volume-based and feature-based. The research presented here proposes a feature-based defeaturing technique which computes the “gross shape” (similar to the original shape but with far lesser details) [10]. It does not take into account the usual CAE inputs such as load paths, boundary conditions, symmetry, etc., thus making it suitable even for other operations like shape matching, retrieval, etc. CAE inputs can also be integrated on top as per the requirements. .

Dabke [11] through the concept of “global idealization”, was one of the first researchers to leverage the feature information for defeaturing. Lee [12] elaborated a technique to reorder features in the history tree and then build them partly up to a given level of simplification. Russ [13] decided suppressibility on the feature type, feature dimensions, proximity of features to the boundary conditions, analysis type and part dimensions. Commercial packages like ACIS®, Autodesk Fusion®, Altair’s Hypermesh®, etc. also provide similar defeaturing capabilities mainly for CAE analysis, but have to do heuristic feature recognition first, which is prone to failures.

## 2.3 Feature Abstraction

Features, not only carry shape information (geometry and topology) but also embed meta-information based on the application context. With a variety of applications, each having their own needs, has resulted into various *feature-schemes*. This makes writing a generic/portable feature-based algorithm, a humongous task. Abstraction alleviates this problem by the extraction of a generic feature-form from various specific features.

Middleditch provided feature abstraction by proposing structure, construction sequence and point-set, so as to separate issues of solid modeling, feature modeling and constraints [14]. Tessier tried to formulate features in terms of generic attributes such reference attributes for reference planes, parameter attributes for model type, depth, etc.[15].

From the research reviewed so far, it appears that there is no feature abstraction schema, generic enough to represent the CAD form features. This research attempts to propose one, by abstracting form features as generic as a Loft feature with profiles and a guide curve.

## 2.4 Cellular Decomposition

Cellular decomposition, in the context of the current research, is the division of a feature-based CAD model into non-volumetrically overlapping, but surface-overlapping sub-solids (called “cells”). 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 [16–18]. There have been a few attempts to compute a midsurface using cellular-decomposition as well [18–21], but the methodologies presented therein appear to be limited due to extensive use of heuristic rules, such as hard-coded values for detecting edge/face pairs, support for limited types of geometries, only limited scope in terms of the range of connections handled, etc.

For example, Woo’s [20] approach, uses face-pairing using distance criteria, which may generate incorrect pairs. It is restricted to only analytical surfaces and only parallel face-pairs.

## 2.5 Analysis of the Survey Findings

After analyzing various past approaches, both in the academic and commercial domains, it was concluded that there has been limited success in the computation of a well-connected midsurface. Commercial packages like Autodesk Inventor®, MSC’s Smart Midsurface®, Altair’s Hypermesh®, etc. provide automatic midsurface functionality, but they appear to result in disconnected midsurface, especially for the non-trivial parts, and require manual patching to get them connected. By far, the most widely-used commercial approach appears to be the Midsurface Abstraction (Face Pairing), but even that also is not devoid of issues. Two of the most critical problems are elaborated below:

**Face Pairs Detection Problem:** In MA, each face-pair computes its own midsurface patch. Finding the face-pairs in a complex part is very challenging [18]. This research addresses this problem by using the feature-information for computation of patches and avoids detection of face-pairs. Feature abstraction, called *ABLE*, has been proposed to generalize various form-features into a generic “Sweep/Loft” feature having a profile and a guide curve. Midsurface patch can then be computed by sweeping midcurve of the profile along the guide curve (more details in section 6.3) [22].

**Face-Pair Interaction Problem:** Midsurface patches need to be connected in a manner similar to how the face-pairs in the input model are connected. As there is no theoretical framework encompassing all the possible interaction types, providing a unified logic has not been possible [4]. This research proposes leveraging of feature-based cellular decomposition to decompose the given solid into sub-volumes and then devise a generic logic for joining the midsurface patches (more details in section 6.5).

## 3 Proposed Approach

Following section provides an overview of the proposed system and the overall work-flow (Fig 4):

- **Input:** This research expects a feature-based CAD (Fb-CAD) model of a thin-walled solid as the input. The model

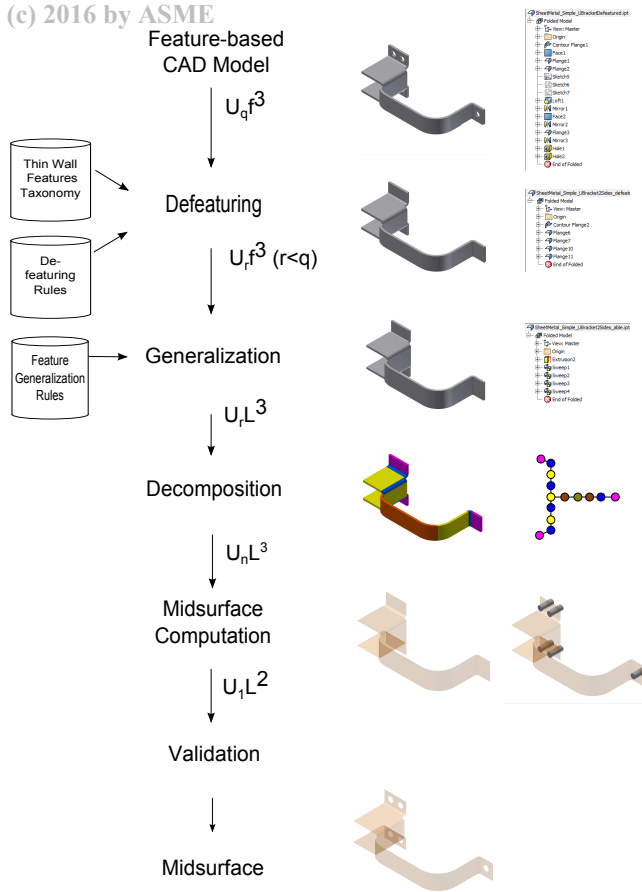


Figure 4: Overall Workflow

is represented by  $(\cup_q f^3)$ , where ‘ $\cup$ ’ denotes collection of ‘ $q$ ’ number of features ‘ $f$ ’, having dimensionality ‘3’, meaning ‘solids’. For non-FbCAD formats such as mesh, solid B-rep, etc., there are techniques such as segmentation, feature recognition (FR), etc. for transformation into FbCAD.

- **Defeating/Simplification<sup>1</sup>:** Computes the gross shape by removing the irrelevant/superficial features. After defeating, the model becomes  $(\cup_r f^3, r \leq q)$ . Number of features are reduced from  $q$  to  $r$  [23] (elaborated more in section 4).
- **Generalization/Abstraction<sup>2</sup>:** *ABLE* (Section 2.5) transforms form-features into the “Loft” representation ( $L$ ) making it simpler to develop further algorithms. The model thus becomes  $(\cup_r L^3)$ , where each feature ‘ $f$ ’ is now replaced by ‘ $L$ ’ [22] (elaborated more in section 5).
- **Decomposition:** Cellular decomposition is performed to form non-volumetrically-overlapping cellular bodies having respective owner-Loft feature. A cellular-graph is populated with the cellular bodies at the graph-nodes and face-interfaces as edges. Number of cells ‘ $n$ ’ may be  $\geq$  than the number of Lofts ‘ $r$ ’ (elaborated more in section 6.1).

<sup>1</sup>Terms “Defeating” and “Simplification” are used synonymously in this work, unless specified otherwise.

<sup>2</sup>Terms “Generalization” and “Abstraction” are used synonymously in this work, unless specified otherwise.

- **Midsurface Computation:** Using topology of the graph, the nodes are classified into midsurface-patch generating nodes (solid cells - *sCells*) and interaction-resolving nodes (interface cells - *iCells*). The output is a single surface (shown by ‘1’ with dimension ‘2’) of the Loft type (elaborated more in section 6).
- **Validation:** Midsurface needs to faithfully mimic the input shape. A topological method is used for this validation [24] (elaborated more in section 7).
- **Output:** A well-connected midsurface is then sent to downstream applications such as CAE.

In comparison with the previous approaches, the proposed one addresses problems (Section 2.5), by effectively leveraging feature information in defeating and abstraction, and thus removes the need of error-prone face-pairing or feature recognition. It utilizes feature-based-cellular decomposition to make connection logic generic by reducing it to type-independent rules. Following sections provide details of each of the above-mentioned modules. This paper presents an aggregate workflow of these modules, which have been individually presented in the past publications [10, 22–27].

## 4 Defeating/Simplification

It has been observed that a simplified shape, which retains the overall shape, called “gross shape”, yields more effective midsurface [23]. Thus, the objective of this defeating module is to compute the gross shape by removing the irrelevant features. The selection/eligibility of the irrelevant features is decided in two phases. In the first, which is sheet metal domain-specific, a newly-proposed taxonomy (Figure 5) is used for deciding suppressibility. In the second, a generic technique based on geometric reasoning is proposed to identify the suppressible features. These are implemented as phases I and II, and appear sequentially, one after the other. Further more, both of these phases can be customized. Phase I can be customized for different domains by providing their respective taxonomies. Phase II can be customized with different size-threshold values.

### 4.1 Phase I

The proposed taxonomy (Figure 5) is used to decide the suppressibility of a sheet metal feature and the process of Phase I is presented in Algorithm 1.

#### 4.1.1 Sheet Metal Features Taxonomy

**Primary Features:** These are not suppressed irrespective of their size as they form the principal/gross shape. Their absence will create missing midsurface patches. Examples: Face-Wall, Flange, Drawing, etc.

**Secondary Features:** These are suppressed based on their relative smaller size with respect to the overall input-shape and the size-threshold. Examples: Stamp-ing, Emboss, etc.

**Tertiary/Auxiliary Features:** They are helper/ancillary shapes and thus, not a part of the gross/overall shape. So they can be suppressed irrespective of their sizes. Examples: Lip, Rest, etc.

**Feature Groups:** These are feature collections. Their suppressibility is evaluated as a collection with the size-based criterion. Examples: Mirror, Patterns, etc.

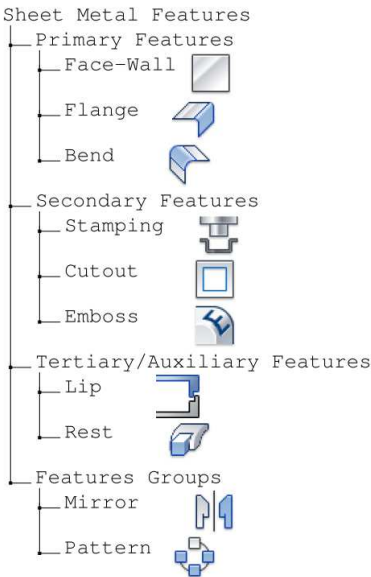


Figure 5: Sheet Metal Features Taxonomy (Icons Source: Autodesk Inventor [28])

Figure 6 shows the process pictorially. The first column, called “Input” shows input sheet metal features CAD model of a bracket part. The second column, called “Selections” shows the candidate features selected, such as ‘Corner Round1’ on the model, based on the steps mentioned above (also mentioned in Algorithm 1). The third column shows the output of Phase I, showing the model with candidate features removed.

Input	Selections	Output

Figure 6: Phase I Results

**Algorithm 1** Phase I: Defeaturing Sheet Metal Features

```
Require: A Sheet Metal FCAD model with access to the feature tree
1:  $p = getPreDefinedThreshold()$ 
2:  $Area = sumAllFaceAreas()$ 
3:  $D = \frac{p}{100} \times Area$ 
4: while  $nextFeature() \neq null$  do
5:    $f_i = currentFeature()$ 
6:   if  $f_i \rightarrow isTertiaryFeature()$  then
7:      $sl \rightarrow add(f_i)$ 
8:   else if  $f_i \rightarrow isGroupFeature()$  then
9:      $Area_{feature} = f_i \rightarrow combinedArea()$ 
10:    if  $Area_{feature} < D$  then
11:       $sl \rightarrow add(f_i)$ 
12:    end if
13:  else
14:     $size\ Area_{feature} = f_i \rightarrow area()$ 
15:    if  $Area_{feature} < D$  then
16:       $sl \rightarrow add(f_i)$ 
17:    end if
18:  end if
19: end while
20:  $sl \rightarrow removeAll()$ 
21:  $update()$ 
```

**4.2 Phase II**

FbCAD model is built step-by-step, using features at each step. Feature parameters are used to compute the “canonical” (tool-body) volume first, which is then booleaned to the model built till then. During this operation, some portion of the canonical volume may get consumed, leaving behind the remaining (remnant) volume in the final solid (Figure 7). Identification of suppressible features based on the feature volume computed from the full feature parameters yields incorrect results [13] as the final shape may not retain the full feature volume. So, this work uses the size of remnant feature volume to decide the suppressibility (Figure 8) of the features.

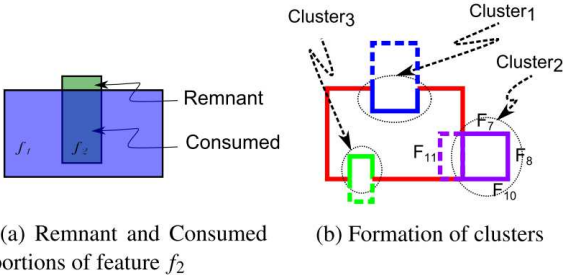


Figure 7: Phase II : Remnant Feature Volumes



Clusters	Size	Feature
cluster <sub>1</sub>	0.25	Extrude <sub>2</sub>
cluster <sub>2</sub>	0.25	Extrude <sub>3</sub>
cluster <sub>3</sub>	0.125	Hole <sub>1</sub>

Table 1: Clusters

Algorithm to identify candidate features for de-featuring based on the Remnant Feature method is presented in Algorithm 2: For each remnant (the one that survives even in the final body) face, its owning feature is extracted via attributes stored on them. Clusters/Groups of such faces are built based on the owning features as shown in Figure 7. The dotted portion in a cluster represents the consumed feature, whereas the encircled portion is the remnant feature. Size of the cluster can be calculated by various techniques like Influence Volume (obtained as a difference of the volume, if the feature is suppressed and then unsuppressed) or the union of bounding-boxes, etc. This work uses summation of the area of the remnant faces (Table 1) as per the Size criterion. Each cluster-owning feature(s) is added for suppression based on the threshold value given by the user.

#### Algorithm 2 Phase II: Remnant Feature Method

**Require:** A CAD model with access to the feature tree.

```

1: while nextFace() != null do
2:    $F_i = \text{currentFace}()$ 
3:    $\text{feat} = F_i \rightarrow \text{owingFeature}()$ 
4:    $\text{addedFlag} = \text{false}$ 
5:   while nextFaceGroup() != null do
6:      $\text{cl}_j = \text{currentFaceGroup}()$ 
7:     if  $\text{cl}_j \rightarrow \text{owingFeature}() == \text{feat}$  then
8:        $\text{cl}_j \rightarrow \text{add}(F_i)$ 
9:        $\text{addedFlag} = \text{true}$ 
10:    end if
11:  end while
12:  if  $\text{addedFlag} == \text{false}$  then
13:     $\text{cl}_n = \text{newFaceGroup}()$ 
14:     $\text{cl}_n \rightarrow \text{owingFeature}() = \text{feat}$ 
15:     $\text{cl}_n \rightarrow \text{add}(f_i)$ 
16:  end if
17: end while
18: while nextFaceGroup() != null do
19:    $\text{cl}_k = \text{currentFaceGroup}()$ 
20:    $\text{size} = \text{cl}_k \rightarrow \text{calculateSize}()$ 
21:   if  $\text{size} < D$  then
22:      $\text{sl} \rightarrow \text{add}(\text{cl}_k)$ 
23:   end if
24: end while
25:  $\text{sl} \rightarrow \text{removeAll}()$ 
26: update()
```

Figure 8 shows the process pictorially. The first column,

called “Input” shows the sheet metal features CAD model of a bracket part, partially defeatured in the Phase I. The second column, called “Selections” shows the candidate features selected, such as Holes, on the model, based on the steps described above (also mentioned in Algorithm 1). The Hole is made by cutting a cylinder-like tool body. This tool body’s dimensions are big enough to disqualify it from removal. But the remnant faces in the model are small enough to qualify this feature for removal. The third column shows the output of Phase I, showing the model with candidate features removed.



Figure 8: Phase II Results

In this work, the effectiveness of defeaturing is computed by measuring **percentage reduction in the number of the faces** ( $pR$ ), while keeping the overall shape intact (%):

$$pR = \left(1 - \frac{rF}{nF}\right) \times 100 \quad (1)$$

Where,

1. Total number of the faces in the original part ( $nF$ )
2. Number of faces left after Phase II ( $rF$ )

Higher the percentage, more effective is the defeaturing process (Section 8.1).

#### 4.3 Dormant Feature-Bodies

In addition to the phases mentioned above, a novel idea of caching large/relevant negative features is used. Large/relevant but negative features like Holes, Cuts, etc. which, by usual rules would not get suppressed, are suppressed and their tool/canonical bodies are preserved. These bodies, called “Dormant bodies” are then used to pierce the final midsurface. With this arrangement, computing midsurface patches is simplified due to lack of holes, while repiercing of the cached dormant bodies ensures their faithful representation in the midsurface (Refer example in Section 8.4 for more details).

#### 5 Abstraction/Generalization

In this module, input feature tree is converted into Loft-equivalents’ tree (Figure 10) [22]. This abstracted tree is then sent for Midsurface computation.

Loft is a generic operator (Figure 9) capable of generating most of the basic shapes. It joins *profiles* along a guide *curve* and represented as  $\Omega_{\mathcal{L}}^{subtype,3}[\{0, curve, 0|C_{0,1,2}\}((sketch)^{<1-n>})]$ . *Continuity* options are  $C_0$  for connectedness,  $C_1$  for tangency and  $C_2$  for curvature continuity. Output of the Loft can either be a *solid* (where capping faces are added to close the shape) or a *surface* (capping faces are not added) and accordingly, dimensionality of 2/3 can be specified<sup>3</sup>.

the threshold-factor ( $t$ ) times the maximum dimensions ( $d_3$ ), denoted as  $d_1 < t.d_3$  &  $d_2 < t.d_3$ .

This classification strategy simplifies the complexity of the midsurface generation problem to a great extent. Since generating a midsurface patch for  $sCell$  is relatively straightforward, these are discrete volumes with known owning-feature of only one type “Loft”. For Loft (or for its equivalents like Sweep, Revolve and Extrude), the midsurface patch can be computed in a deterministic manner. The problem of resolving interaction amongst the midsurface patches is carried out in the  $iCells$ , again in a reliable manner.

Advantage of the CAG representation is that it is easier to classify nodes and delegate specific computational work to them. Actual computation of midsurface geometry is delegated to the nodes having unique owning-feature, where as others are meant to join the midsurface patches. Thus, this research does not need to enumerate specific heuristic rules used for specific connection types. Midsurface computation strategies for  $sCells$  and  $iCells$  are detailed out in the sections below.

### 6.3 Computing Midsurface Patches in $sCells$

The CAG is traversed node by node. The owning-feature corresponding to the cell represented by the current node is extracted. A midsurface patch is computed looking at the feature parameters of owner Loft feature, such as the profile  $p$  and the guide curve  $g$  [22].

For each Sweep<sup>4</sup> feature, the midsurface patch can be generated in the following manner:

**Thin Profile:** If *profile* is very small as compared to *curve* ( $profile_{length} \ll curve_{length}$ ), then *midcurve* (Section 6.4) is extracted from the *profile* and swept along the same *curve*. This rule is expressed as  $\Omega L^{L,2}[\{0, curve, 0 | C_{0,1,2}\}(midcurve^{1-n})]$ . **Midcurve** is a set of *curves* lying midway of a 2D *profile* and is expressed as  $\Omega M^{C,1}[\{\}(profile)]$ .

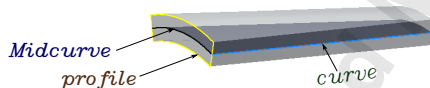


Figure 12: Thin Profile

**Thin Loft :** If *profile* is very big as compared to *curve* ( $profile_{length} \gg curve_{length}$ ), then *midcurve* is not extracted, but the *profile* itself is swept along half of the *curve*. This rule is expressed as  $\Omega L^{L,2}[\{0, curve/2, 0 | C_{0,1,2}\}(profile)]$ .

**Thick Sketch:** If *sketch* is comparable in size to *curve* ( $sketch_{length} \approx curve_{length}$ ), then it is a thick shape and Midsurface is not generated.

<sup>4</sup>Sweep and Loft, though distinct, are used interchangeably here in the context of constant thickness thin-walled solids.

Applying these strategies for  $sCells$  shown in Figure 11, the corresponding midsurface-patches generated are shown in Figure 13.

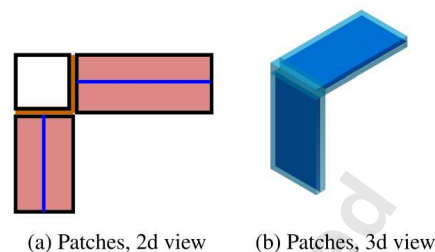


Figure 13: Midsurface Patches

#### Algorithm 3 $sCell$ Midsurface Patch Computation

**Require:**  $sCell$

**Ensure:**  $size(sCell \rightarrow owning - feature(s)) = 1$

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

At this stage, the  $sCells$  are filled with the midsurface patches and are ready to get connected in  $iCells$  (section 6.5). Following section details the process to compute midcurve needed in case of a Thin-Profile.

### 6.4 Midcurve

Midcurve is a set of connected 2D curves lying midway of a 2D profile. This research focuses on 2D planar polygonal sketch profiles with an assumption that curved shapes can be converted to polygonal shapes by faceting. In the first phase of the computation, a polygon is partitioned into primitive sub-polygons and then in the second phase, midcurve-segments are computed in the non-interface-polygons. At the end, the individual midcurve-segments are extended-joined to form a continuous set of curves mimicking the parent shape.

#### 6.4.1 Polygon Decomposition

A polygon can be partitioned into convex regions by eliminating all reflex (concave) vertices. More details are elaborated in [26] [27]. This technique improves upon the Bayazit's algorithm [32] in terms of expanding search to include even extreme vertices in the range, thereby giving minimal and elongated partitions.



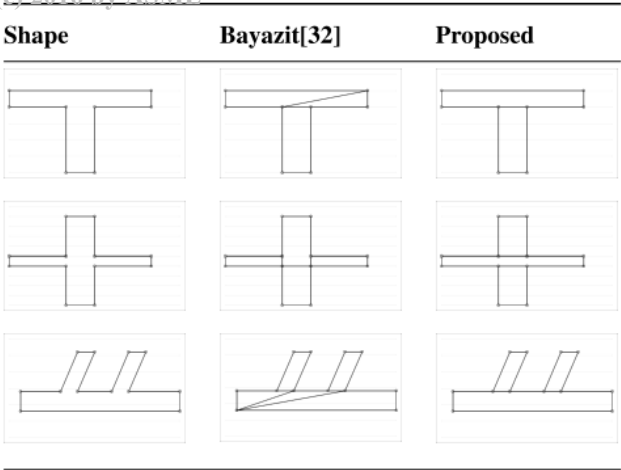
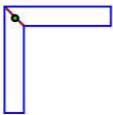


Figure 14: Improvement over Bayazit’s Algorithm

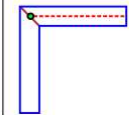
The resulting sub-polygons are sent for creating connected midcurves. Decomposition helps in getting the sub-polygons which are of primitive types and which are easier for midcurves’ creation as compared to the original whole shape.

6.4.2 Midcurve Computation

Each partitioning edge inserted during the decomposition is called as a ‘chord’.



Midcurves are generated for individual polygons that are longer length-wise on both sides of the ‘chord’.



When curves do not meet at the chord, they are extended upto the midcurve.

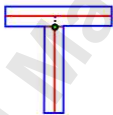


Figure 15: Midcurves Segment Joining

Following are the sample profiles taken from past research papers, and along with the midcuves computed from the above-mentioned process.

6.5 Resolving Interactions in *iCells*

Input to this module is a CAG with midsurface patches computed at all the *sCells*. The sole responsibility of *iCells* is to connect the midsurface patches incident on them, either by extending the midsurface-patches from the adjacent *sCells*, or by generating new ones. The CAG is used to traverse *iCells* one by one. For each *iCell*, the midsurface patch interactions are resolved as follows (Algorithm 4).

Each *iCell* is connected with adjacent cells via edges

Glass profile by Fischer [33].	Profile by Ramanathan [6].	Plastic Part profile by Sheen [34].

Figure 16: Midcurves Benchmarking

*e*(*s*). Each incident *e* has two nodes, 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 (Figure 17b). An intersection curve ( $m^1$ ) is computed between the overlapping face ( $O_{1,2}$ ) and the midsurface ( $m^2$ ). This curve acts as a midcurve for the extension into the *iCell*. In case, where all the three cells are geometrically flat and relatively big, all are marked as *sCells* (not *iCells*) and have no extensions.

If the “adjacent node” is an *iCell*( $n_3$ ), then an extra patch is created between the centroids of the two *iCells* (Figure 17c). Thus, all the *iCells* join the midsurface patches (Figure 17a) to form a well-connected midsurface. Figure 18 shows *iCell* – *iCell* resolution in 3D.

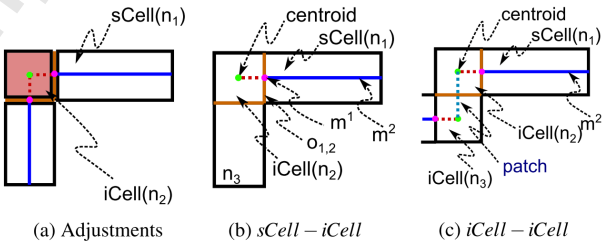


Figure 17: Resolving Interactions in the *iCell*

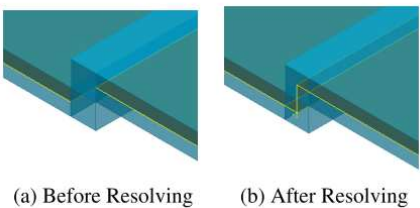


Figure 18: *iCell* Resolving in Overlap Case

Once all the interactions are resolved in *iCells*, the result is a well-connected midsurface. Cached dormant bodies

**Algorithm 4** *iCell* Midsurface Patch Interaction Resolution

```

Require: iCell
1: while size(iCell → 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
    
```

are pierced into the midsurface (Section 4.3) to restore the temporarily suppressed negative features (Figure 19).

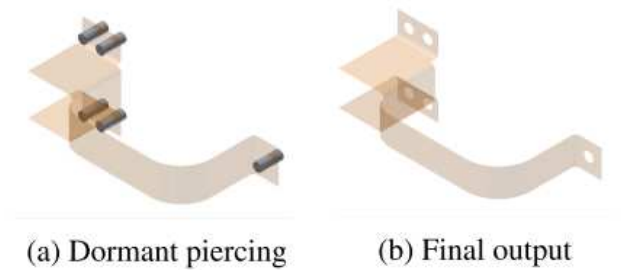


Figure 19: Computation of Midsurface of a Bracket

Table 20 shows various academic cases, with their respective cellular representations, graphs and midsurface outputs. Table 20 demonstrates the range of shapes and connection types handled by the proposed approach. **L**, **T**, **Plus** cases show how 2,3 or more primitives coming at a junction form proper connections. **Curved Y** shows non-planar midsurface patches joining with a planar one. **Wavy Tri** shows non-analytic sub-shapes shapes case working properly. **Drafted Plus** has some of the sub-shapes with variable thickness, i.e. non-parallel pairs. **Overlap**, **LPlus** cases are mentioned in Woo’s [20] work are demonstrated to work fine here. It appears that his and few others’ [19, 21, 33] approaches may face difficulties in shapes having **Wavy Tri** and **Drafted Plus** cases.

### 7 Validation

Midsurface is expected to truly represent the original solid, both in terms of geometry and topology. It is tedious to manually inspect and validate it, especially for the complex models. Apart from geometrical validation, it is nec-

Name	Part	Cellular	Graph	Result
L				
T				
Curved Y				
Wavy Tri				
Plus				
Drafted Plus				
Overlap				
LPlus [20]				

Figure 20: Feature-based Cellular Midsurface

essary to carry out topological validation to verify the connectivity as well as missing surfaces or gaps. The following section demonstrates the formulation developed to carry out such validations [24].

The main idea behind the formulation is derived in two ways. In the first method, the generated midsurface is expected to be such that it is obtained as if the solid body is shrunk to its overall neutral plane. In the second method, as a reverse validation, if such a midsurface is thickened, it should represent a solid body as close to the original solid as possible. The solid’s topological validity is then checked using Euler-Poincaré’s manifold equation. If the original midsurface had any gaps or overlaps, the resultant solid will not satisfy the manifold condition. The proposed procedure to validate a midsurface is as follows (more details in [24]):

Steps:

1. Classify Midsurface entities as :  $f, e_s, e_{sr}, e_{rr}, e_r, e_i, v_s, v_r, v_i, s, h, r$  (Figure 21).

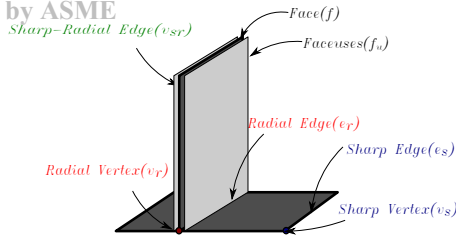


Figure 21: Non-Manifold Topological Entities

2. Predict topological entities using the equations developed [24].
3. Verify that the predicted topological entities of the thin-walled solid satisfy the manifold equation by showing that left ( $\chi_{ml}$ ) and right ( $\chi_{mr}$ ) hand side of the equation matches.

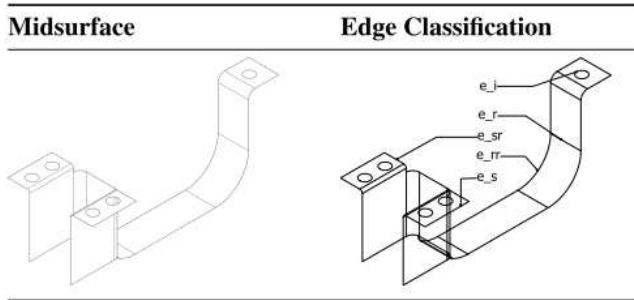


Figure 22: Edge Classification

1. **Midsurface entities:**  
 $f = 15, e_s = 3, e_{sr} = 10, e_r = 14, e_{rr} = 19, l_p = 9, e_i = 5, v_s = 8, v_r = 24, v_i = 5, s = 1, h = 5, r = 5$
2. **Predicted solid-faces:**  
 $f_m = 2f + e_s + l_p + e_i$   
 $= 2 \times 15 + 3 + 9 + 5 = 47$
3. **Predicted solid-edges:**  
 $e_m = 2(e_s + e_{sr} + e_{rr} + e_i) + \sum n_r e_r + v_s + v_i$   
 $= 2(3 + 10 + 19 + 5) + (2 \times 12 + 4 \times 2) + 8 + 5 = 119$
4. **Predicted solid-vertices:**  
 $v_m = 2(v_s + v_i) + \sum n_r v_r$   
 $= 2 \times (8 + 5) + 2 \times 24 = 74$
5. **Predicted solid-shells-holes:**  
 $s_m = s = 1, h_m = r_i = 5, r_m = 2r_i = 10$
6. **Manifold equation of the left side ( $\chi_{ml}$ ):**  
 $= v_m - e_m + f_m$   
 $= 74 - 119 + 47 = 2$
7. **Manifold equation of the right side ( $\chi_{mr}$ ):**  
 $= 2(s_m - h_m) + r_m$   
 $= 2(1 - 5) + 10 = 2$

It can be observed that the predicted solid entities validate the manifold equation ( $\chi_{ml} = \chi_{mr} = 2$ ) and the topological entities of the thin-walled solid match with the predicted ones.

## 8 Results

Objective of this work is not only to devise and implement a collection of algorithms for computing the mid-surface, but also to test and assess the output on real life sheet metal parts. “Enclosure” is a typical sheet metal casing model used in electronics equipments. It houses circuits, wires, fan, etc.

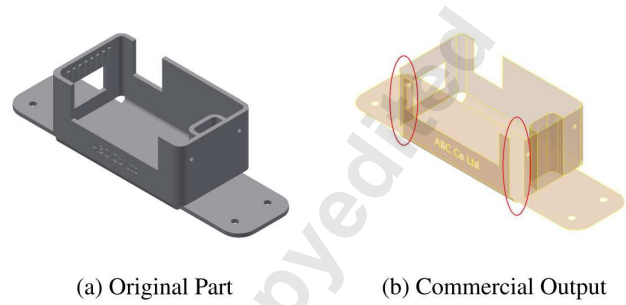


Figure 23: Comparison of Midsurface Outputs

Many failures are observed such as disconnected patches, missing midsurface patches, etc. Following sections show progress of the part, while going through the various modules (Figure 23).

### 8.1 Simplification

Simplification/defeaturing removes unwanted features to compute the “gross shape”. It also caches tool-bodies of non-suppressible negative features to be used for piercing after midsurface computation. Threshold (D) used as a size threshold here is 5% of the total part size.

Effectiveness with 5% threshold, per Eqn. 1 is:

Qty	Input	Phase I	Output
Faces	259	104	64
Suppressed		13	8
$pR = (1 - \frac{64}{259}) \times 100 = 75.29\%$			

Even after a substantial reduction in the number of faces (75%), the overall shape of the enclosure is retained fine.

### 8.2 Abstraction

The purpose of this module is to transform sheet metal feature tree into an  $\mathcal{ABLE}$  feature (Extrude, Revolve, Sweep, Loft) tree. Effectiveness of this module depends upon the faithful reproduction of the part without any feature or shape loss.

### 8.3 Decomposition

The given abstracted part is decomposed at feature level as well as at convex edges to form sub-volumes/cells.

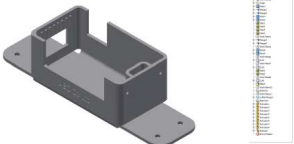
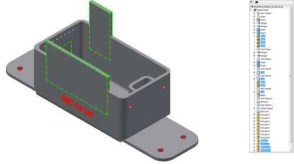
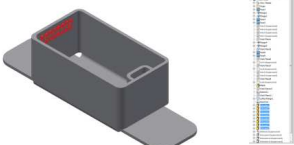
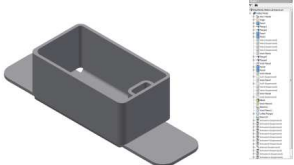
Model	Tree	Explanation
Original/Input		The model with full CAD details.
Phase I Selections		Small holes, embossing are chosen based on Sheet Metal feature taxonomy.
Phase II Selections		Remnant features get selected in the second phase.
Defeatured		Most of the unnecessary features are removed.

Figure 24: Defeating

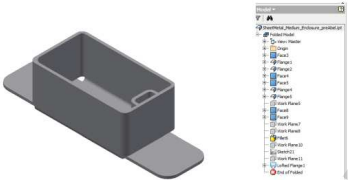
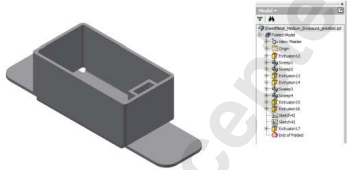
Model	Tree	Explanation
Defeat Input		The model has Sheet metal features such as FACE, Flange, Lofted Flange, etc.
Abstract Output		All Sheet Metal features are abstracted.

Figure 25: Abstraction

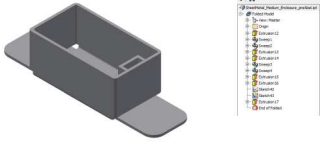
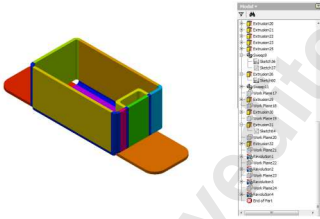
Model	Tree	Explanation
Abstract Input		Internal booleans of Primitive features are set to "New", Partitioning at convex edges is done.
Decomp Output		List of sub-volumes/cells with a primitive/ <i>ABLE</i> owner feature assigned.

Figure 26: Decomposition

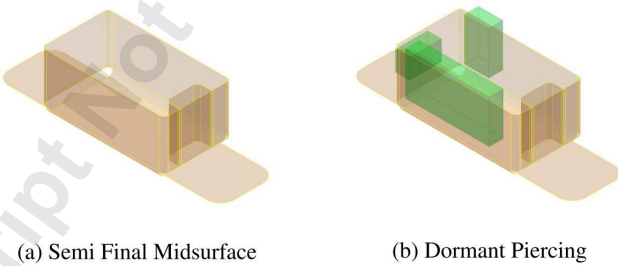


Figure 27: Midsurface Computation

The final midsurface is shown in Figure 28.

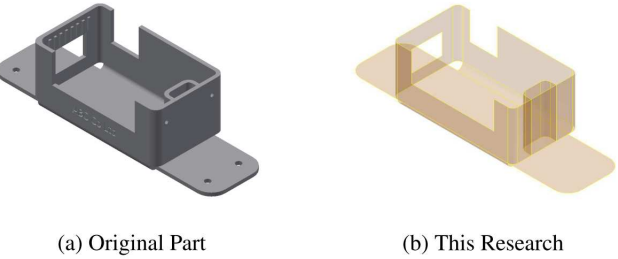


Figure 28: Input and Output Comparison

8.4 Computation

Midsurface patches and connections are computed to result into a well-connected midsurface of the gross shape. Dormant bodies cached during defeating module are brought back to pierce into this midsurface, so as to generate the pending cuts (Figure 27b).

8.5 Test Cases

Following additional test cases (Figure 29 and Figure 30) demonstrate efficacy of the approach with few more practical examples.



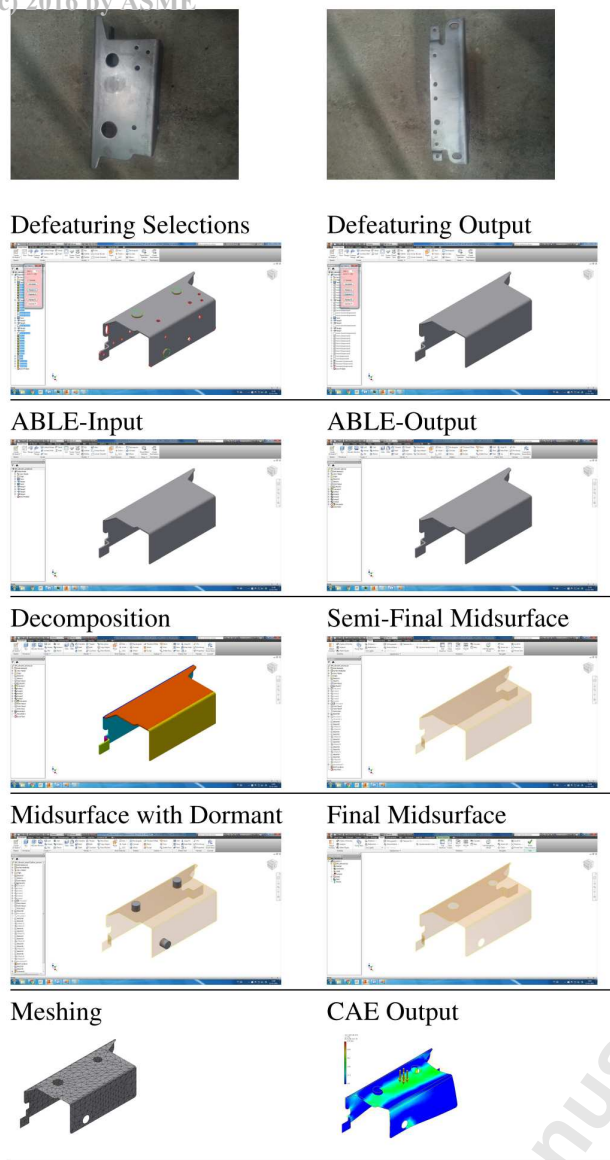


Figure 29: The Rectangular Clip Bracket Part

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 amongst 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

Computation of midsurface is one of the most popular and important idealization techniques for CAE analysis of thin-walled models. Study of the existing techniques revealed critical insights into the problems of the midsurface results (gaps, overlaps, lack of isomorphism, etc.), such as failures in the face-pair detection and interactions.

All the sub-processes used in this work leverage feature-

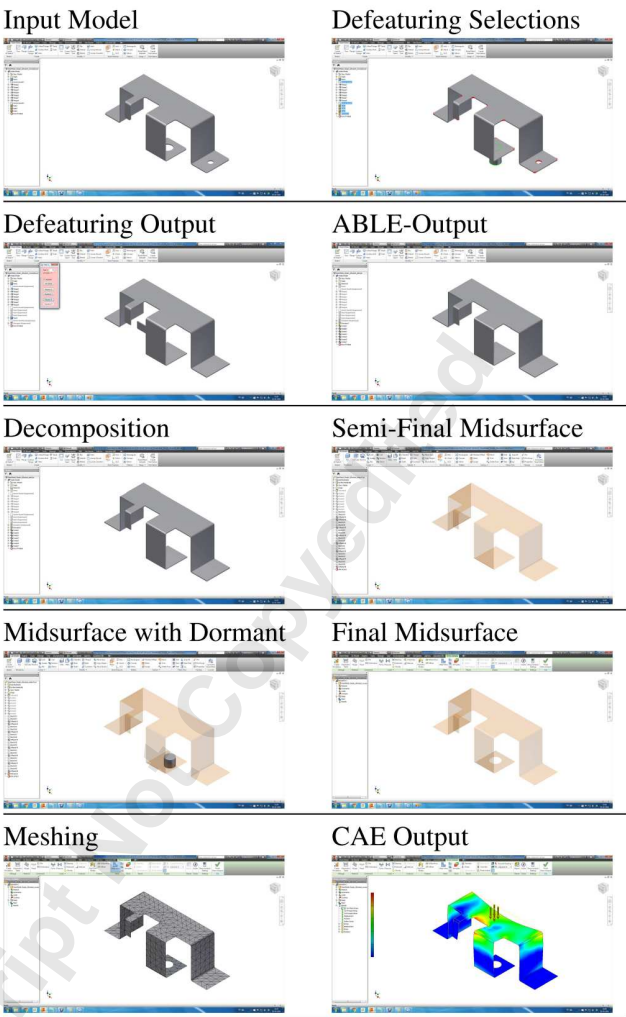


Figure 30: Input The U Bracket Clip Part

information effectively, thereby introducing a new way of **FSAD** (Feature-based, Simplification, Abstraction and Decomposition) methodology to CAD algorithms.

The superiority of the proposed approach is the manner in which it has abstracted both the sub-problems. Following are the salient contributions of this approach with respect to past ones:

- **Simplification:** Domain-specific taxonomies can be introduced for defeaturing to suit particular applications. The generic Remnant feature approach is far accurate than full-feature parameters, for detection of suppressible features.
- **Abstraction:** Instead of devising algorithms for a plethora of feature shapes, a singular Loft-based feature-transformation is far more effective as well as maintainable. Any new introduction of a domain specific feature, just needs one-time transformation rule; rest of the midsurface algorithm works without any changes.
- **Decomposition:** Divide-and-conquer methodology, reduces the complexity of the problem. Cells, in this case, the loft primitives, are easy to create midsurface patches



and the graph structure of the cells gives accurate information needed to address midsurface connectivity issues.

Although individual functionalities such as defeaturing, abstraction and decomposition processes may show certain shortcomings, but overall, the proposed approach seems to work well and is able to compute a well-connected midsurface in a robust, definitive and generic manner, with minimal failures.

## References

- [1] Li, M., Zheng, J., and Martin, R. R., 2012. "Quantitative control of idealized analysis models of thin designs". *Computers and Structures*.
- [2] Aparicio, C., 2015. 3 steps to create midsurface geometry, Sept. (Last accessed Oct 2015).
- [3] Robinson, T. T., Armstrong, C. G., McSparron, G., Quenardel, A., Ou, H., and 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.
- [4] Stolt, R., 2006. "Reusing cad models for die-casting products for fea". *Proceedings of 2nd NAFEMS Seminar: Prediction and Modelling of failure using FEA*, May.
- [5] Lockett, H., and Guenov, M., 2008. "Similarity measures for mid-surface quality evaluation". *Comput. Aided Des.*, **40**(3), Mar., pp. 368–380.
- [6] Ramanathan, M., and Gurumoorthy, B., 2004. "Generating the mid-surface of a solid using 2d mat of its faces". *Computer Aided Design and Applications*, **1**, pp. 665–674.
- [7] Blum, H., 1967. "A transformation for extracting new descriptors of shape". In Proc. Models for the Perception of Speech and Visual Form, W. Wathen-Dunn, ed., MIT Press, pp. 362–380.
- [8] Rezayat, M., 1996. "Midsurface abstraction from 3D solid models: general theory, applications". *Computer-aided Design*, **28**, pp. 905–915.
- [9] Thakur, A., Banerjee, A. G., and Gupta, S. K., 2009. "A survey of CAD model simplification techniques for physics-based simulation applications". *Computer-aided Design*, **41**, pp. 65–80.
- [10] Kulkarni, Y., Sahasrabudhe, A., and Kale, M., 2013. "Strategies for using feature information in model simplification". In CAE IIT Madras.
- [11] Dabke, P., Prabhakar, V., and Sheppard, S., 1994. "Using features to support finite element idealizations". *Computers in Engineering*, **1**, pp. 183–183.
- [12] Lee, S. H., 2005. "A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques". *Computer-aided Design*, **37**, pp. 941–955.
- [13] Russ, B., 2012. "Development of a cad model simplification framework for finite element analysis". *Master's Thesis, University of Maryland (College Park, Md.)*.
- [14] Middleditch, A., and Reade, C., 1997. "A kernel for geometric features". In Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications, SMA '97, ACM, pp. 131–140.
- [15] Sean Tessier, Y. W., 2013. "Ontology-based feature mapping and verification between cad systems". *Advanced Engineering Informatics*, **27**, pp. 76–92.
- [16] Bidarra, R., and Teixeira, J. C., 1993. "Intelligent form feature interaction management in a cellular modeling scheme". In Symposium on Solid Modeling and Applications, pp. 483–485.
- [17] Woo, Y., 2003. "Fast cell-based decomposition and applications to solid modeling". *Computer-aided Design*, **35**, pp. 969–977.
- [18] Boussuge, F., Léon, J.-C., Hahmann, S., and Fine, L., 2014. "Extraction of generative processes from b-rep shapes and application to idealization transformations". *Comput. Aided Des.*, **46**, Jan., pp. 79–89.
- [19] Chong, C. S., Kumar, A. S., and Lee, K. H., 2004. "Automatic solid decomposition and reduction for non-manifold geometric model generation". *Computer-aided Design*, **36**, pp. 1357–1369.
- [20] Woo, Y., 2014. "Abstraction of mid-surfaces from solid models of thin-walled parts: A divide-and-conquer approach". *Comput. Aided Des.*, **47**, pp. 1–11.
- [21] Zhu, H., Shao, Y., Liu, Y., and Li, C., 2015. "Mid-surface abstraction for complex thin-wall models based on virtual decomposition". *International Journal of Computer Integrated Manufacturing*, pp. 1–18.
- [22] Kulkarni, Y., Sahasrabudhe, A., and Kale, M., 2014. "Formulating midsurface using shape transformations of form features". In AIMTDR.
- [23] Kulkarni, Y. H., Gupta, R. K., Sahasrabudhe, A., Kale, M., and Bernard, A., 2015. "Defeaturing sheet metal part model based on feature information". *Proceedings of CAD'15*, pp. 297–302.
- [24] Kulkarni, Y. H., Sahasrabudhe, A., and Kale, M., 2015. "Topological validation of midsurface computed from sheet metal part". *Computer-aided Design and Applications*, **6**.
- [25] Kulkarni, Y., Sahasrabudhe, A., and Kale, M., 2013. "Using features for generation of midsurface". In Proceedings of International Conference on Advances in Mechanical Engineering (ICAME), pp. 141–145.
- [26] Kulkarni, Y., Sahasrabudhe, A., and Kale, M., 2014. "Midcurves generation algorithm for thin polygons". In National Conference on Emerging Trends in Engineering and Science (ETES), pp. 76–82.
- [27] Kulkarni, Y., Sahasrabudhe, A., and Kale, M., 2017. "Dimension-reduction technique for polygons". *International Journal of Computer Aided Engineering and Technology*. Inderscience.
- [28] Autodesk, 2014. Autodesk inventor 2014 help. Tech. rep., Autodesk. (Last accessed Oct 2015).
- [29] Kageura, M., and Yokohoma, S., 2009. Apparatus and method for generating analysis model.
- [30] Cao, W., Chen, X., and Gao, S., 2011. "An approach to automated conversion from design feature model to

an analysis feature model”. *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, **5**, pp. 655–665.

[31] Woo, Y., 2009. “Automatic simplification of solid models for engineering analysis independent of modeling sequences”. *Journal of Mechanical Science and Technology*, **23**, pp. 1939–1948.

[32] Bayazit, M., 2013. Polygon decomposition. <http://mnbayazit.com/406/bayazit> (Last accessed Oct 2015).

[33] Boussuge, F., 2006. “Idealization of cad assemblies for fe structural analyses”. PhD thesis, University of Grenoble.

List of Tables

1 Clusters . . . . . 6

List of Figures

1 Midsurface with Errors (Source: [2]. Image, Copyright, MSC Software Corporation. Used with Permission.) . . . . . 2

2 Expectations about Midsurfaces . . . . . 2

3 Medial Computation Techniques . . . . . 2

4 Overall Workflow . . . . . 4

5 Sheet Metal Features Taxonomy (Icons Source: Autodesk Inventor [28]) . . . . . 5

6 Phase I Results . . . . . 5

7 Phase II : Remnant Feature Volumes . . . . . 5

8 Phase II Results . . . . . 6

9 Generic Loft Feature . . . . . 7

10 Abstraction/Generalization . . . . . 7

11 Feature-based Cellular Decomposition . . . . . 7

12 Thin Profile . . . . . 8

13 Midsurface Patches . . . . . 8

14 Improvement over Bayazit’s Algorithm . . . . . 9

15 Midcurves Segment Joining . . . . . 9

16 Midcurves Benchmarking . . . . . 9

17 Resolving Interactions in the *iCell* . . . . . 9

18 *iCell* Resolving in Overlap Case . . . . . 9

19 Computation of Midsurface of a Bracket . . . . . 10

20 Feature-based Cellular Midsurface . . . . . 10

21 Non-Manifold Topological Entities . . . . . 11

22 Edge Classification . . . . . 11

23 Comparison of Midsurface Outputs . . . . . 11

24 Defeaturing . . . . . 12

25 Abstraction . . . . . 12

26 Decomposition . . . . . 12

27 Midsurface Computation . . . . . 12

28 Input and Output Comparison . . . . . 12

29 The Rectangular Clip Bracket Part . . . . . 13

30 Input The U Bracket Clip Part . . . . . 13