DEVELOPMENT OF ALGORITHMS FOR GENERATING CONNECTED MIDSURFACES USING FEATURE INFORMATION IN THIN-WALLED PARTS

A THESIS SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY

FOR THE AWARD OF DEGREE OF

DOCTOR OF PHILOSOPHY (PH.D)

IN

MECHANICAL ENGINEERING

SUBMITTED

BY

YOGESH H. KULKARNI

UNDER THE GUIDANCE OF

DR. ANIL SAHASRABUDHE AND DR. MUKUND KALE

RESEARCH CENTER

COLLEGE OF ENGINEERING , PUNE

DEPARTMENT OF MECHANICAL ENGINEERING

AUGUST 2016

To my parents...

Abstract

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. A midsurface is a surface lying midway of (and representing) the input shape. Computation of the midsurface is still a time-consuming and mostly, a manual task due to lack of robust-automated approaches. Many of the existing automatic midsurface generation approaches result in some kind of failures such as gaps, missing patches, overlapping surfaces, etc. It takes hours or even days to correct such errors with manual intervention. The widely used Boundary Representation (Brep) based approaches are computationally intensive, yet cannot guarantee flawless midsurface and are mostly developed for a limited variety of geometric and topological configurations. In these approaches no feature information is efficiently leveraged. Thus, there exists a need to take a holistic look at the CAD model with its feature information and devise a set of efficient algorithms to generate flawless, well-connected midsurfaces capable of handling wide variety of geometric and topological configurations and are also computationally inexpensive.

This thesis is primarily aimed at addressing this need. It provides an integrated approach to address the critical aspects of generation of midsurface for feature based sheet metal CAD model through design and implementation of an intelligent system, called **MidAS** (**Mid**surface **A**lgorithms for **S**heet-metal-parts). It uses CAD models built using Autodesk Inventor and its Application Programming Interfaces (APIs) are used to interact with the model. The algorithms for various modules are implemented in VB and C# .Net programming languages.

The thesis begins by providing an overview of CAD-CAE process, relevance of midsurface in case of CAE analysis of thin-walled parts and describing motivation of choosing the topic of midsurface computation for research. It reviews traditional approaches for generating midsurfaces, comments on their advantages and limitations and brings out the critical gaps. Research objectives are laid down based on the literature review and gap analysis. Then the overview of proposed **MidAS** is presented.

It begins with defeaturing of the input CAD model. Irrelevant features are removed based on criteria such as sheet metal feature type, size of remnant feature portions. Apart from this, bulk negative features, called “dormant” features, are removed tem-

porarily so as to simplify the CAD model without compromising on the gross shape. These dormant features are later reapplied on the generated midsurface.

The defeatured model is further simplified by transforming the remaining sheet metal features into generalized features, representing variations of Loft feature. This generalized representation, called "$\mathcal{ABLE}$ (**A**ffine transformation, **B**ooleans, **L**ofts and **E**ntities)" helps simplify the model further so that generic algorithms can be developed for generating midsurface.

The $\mathcal{ABLE}$ model is simplified further by cellular decomposition. The cells are classified into solid and interface cells. They are delegated with the tasks of creating midsurface patches and joining them, respectively. Midsurface patches are created either by offsetting the profile face or by lofting midcurves of the profiles along the guide of the owner loft feature. Midcurve of the profile is computed by first, approximating it to polygon, then decomposing the polygon and finally, generating midcurves for each sub-polygons. Midsurface patches are joined in the interface cells by a generic logic to form a connected midsurface. The quality of output midsurface is assessed by a topological validation method which is proposed in this research. Towards the end, capabilities of **MidAS** are demonstrated with real-life sheet metal part models. The thesis concludes by highlighting the major contributions and directions for the future work.

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

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Dr. Anil Sahasrabudhe and Dr. Mukund Kale for their guidance and continuous support during my Ph.D research and at the time of writing of this thesis. I could not have imagined having better advisors for this journey.

I would also like to thank the rest of my thesis committee: Prof. Vipin Tripathi, Prof. Arati Mulay, and Dr. S. N. Sapali, for their encouragement.

Last but not the least, I would like to thank my family: my parents, my wife - Anjali, and my kids - Reeya & Deeya, for supporting me in this endeavor.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

These days, industries worldwide are facing several challenges due to stiff global competition. These include rapid changes in the customer preferences, shorter products lives, continuous changes due to disruptive innovations, frequent design changes, etc., forcing industries to get their products in the market, quickly. Time needed from conceptualization of the product till delivering it in market, is known as "Time to Market". Thus, goal of the industries is to shorten the Time to Market by making Product Development Process (PDP) more efficient.

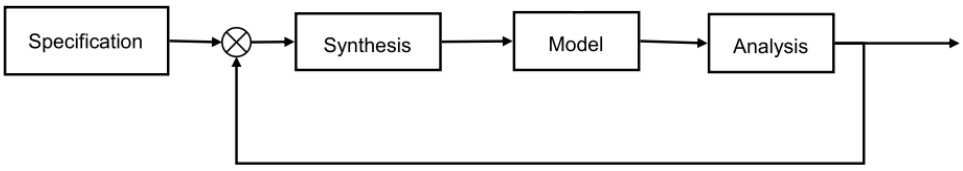


Figure 1.1: Product Development Process (PDP) (Source: Stolt [1])

Figure 1.1 shows stages of PDP as presented by Stolt [1]. It starts with "Specification" stage where the product requirements are studied, prioritized and are turned into concrete specifications. At "Synthesis" stage, various alternatives to address the specifications are evaluated. The most satisfactory alternative is designed at "Model" stage. Finally the designed model is evaluated against the specifications at the ''Analysis" stage. If the analysis results are not satisfactory, then the process is iterated from "Synthesis" stage again, until a satisfactory product is arrived at.

PDP can be made more efficient either by reducing number of iterations or by reducing the time needed for each iteration, i.e. the time needed for the design and analysis of the product. In past, before advent of computers, the design was done by manual calculations, sketches, etc., whereas the analysis was performed by testing physical prototypes of the product. In modern times, Computer-aided Design (CAD) applications

are used extensively for design, whereas Computer-aided Engineering (CAE) applications are used for testing & simulation of the products, virtually. Design calculations provide shape and size of the product. These shapes are modeled in CAD using various operations, such as extruding a sketch, revolve, fillet, etc. These modeling operations are known as features. The resultant CAD models are used for various downstream applications such as Computer-aided Manufacturing (CAM), visualizations, CAE, etc. In CAE, a CAD model is analyzed by decomposing it into mesh of elements, applying loads and boundary conditions and then solving the mesh for parameters such as stress, strain, displacements, etc. Thus, CAD-CAE process is a core stage of the modern PDP process and making it more efficient is key to reduce the Time to Market.

1.2 CAD-CAE Process

One of the critical aspect of the CAD-CAE process is the transformation of CAD model to CAE model. CAD models are often highly detailed, as they need to have information needed by various downstream applications. For CAE, many such details are not needed, because they add to the complexities in mesh generation and demand more computational power-time, for analysis. Removing such details and simplifying the CAD model is a necessary transformation and is a significant portion of preprocessing done to prepare the CAE model. This pre-processing is often not automatic and straightforward, but time consuming, manual and tedious. Thus there is a need to address these problems by a robust and automated simplification process [31].

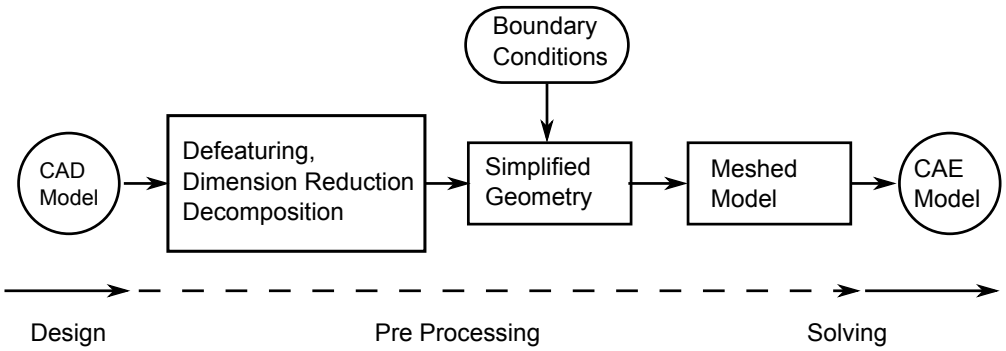


Figure 1.2: Computer-aided PDP (Source: Tierney [2])

Figure 1.2 shows various stages which a CAD model undergoes to be ready for CAE analysis. Simplification, as shown in the first box, mainly consists of defeaturing and dimensions reduction. In defeaturing, irrelevant details are removed by removing corresponding features. Due to defeaturing, meshing is not unnecessarily dense and complicated around the irrelevant details. Thus defeatured model reduces number of elements, which correspond to lesser degrees of freedom (DoFs) of the mesh, thereby

saving analysis time and resources substantially. In dimension-reduction, shapes like slender-bar, thin-wall are transformed into their lower dimension representations such as curves and surfaces, respectively. Lower dimension representations are used for lower dimension element types such as beams or shell which have far lesser DoFs than the solid elements. Thus use of dimension reduction reduces analysis time and resources further.

Dimension reduction transformation is widely used in CAE analysis of thin-walled parts, such as sheet metal and plastic products. CAD models of these parts are transformed into a representative surface known as "Midsurface". It is a surface lying midway of a thin-walled CAD model and mimicking its shape. Getting a correct midsurface is critical to CAE analysis of thin-walled parts.

1.3 Midsurface for CAE Analysis of Thin-walled Parts

CAE analysis of the thin-walled parts using 3D solid mesh elements such a Hexahedral or Tetrahedral, is expensive in terms of computing resources and time taken. 2D surface mesh elements such as Shell, are preferred as they give reasonably accurate analysis results, while requiring far lesser computational resources and time [32]. Shell elements need midsurface and thickness values for their definition and usage. Thus, midsurface is the most widely used representation of thin-walled parts for CAE analysis.

The midsurface is expected to be "well-connected" (patches form a continuous surface) and resembles shape of the input CAD model (more details at Fig. 2.2).

Most midsurface generation approaches are based on the final shape of the CAD model. It is typically represented by a data structure known as Boundary Representation (Brep). In Brep, a solid shape is represented by a set of connected faces to form a closed volume. Midsurface is computed either by applying geometric transformations or by using heuristic rules on Brep.

Face pairing is one of the most popular midsurface computation method based on Brep. Faces opposite each other are detected to form face-pairs. Each face pair computes an equidistant surface in the middle, called midsurface patch. These patches are then, either extended or trimmed to join at a common edge to form a well-connected midsurface. Although this approach works well on simple, academic models, it often fails in case of real-life, complex models. Failures are in the form of gaps between patches, overlaps, midsurface not lying midway, etc. Thus, minimizing the failures and devising a robust approach to compute a well-connected midsurface is critical to the CAD-CAE process of thin-walled parts.

1.4 Motivation of Research

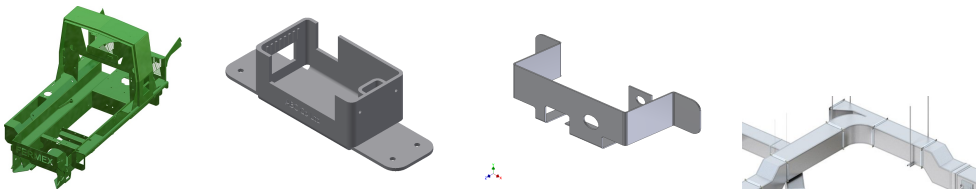
Midsurface is the most suitable representation for the CAE analysis of thin-walled parts' CAD models. A Sandia report [33] states that, for the complex engineering models, their simplification amounts to about 60% of the overall analysis time, whereas the mesh generation consumes about 20% and solving 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 [30]. Automex [34] observed that the complexity involved in generating midsurfaces takes about 70 to 90% of the pre-processing time. Thus, computing well-connected midsurface is highly critical to the industry.

Despite extensive research, many midsurface computation approaches fail to generate a well-connected midsurface, especially for the complex part shapes. Correcting the 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 [15].

Thus, looking at the criticality, the present research focuses on design and implementation of a system to generate a well-connected midsurface for thin-walled parts.

1.5 Scope of the Work

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. D. W. Brown Report [31] suggests that sheet metal parts is one of the most widely used sub-category, with approximately 40% share amongst the manufacturing processes. They are found in a wide variety of applications such automobile, aircraft, shipbuilding, consumer products, electronic equipments, etc.



(a) Harvester (b) Enclosure (c) Bracket (d) Duct

Figure 1.3: Variety of Sheet Metal Parts

Figure 1.3 shows some of the sheet metal parts, such as an agricultural harvester frame [35], an enclosure commonly used in electrical systems, a bracket found in mechanical systems and HVAC ducting. Looking at the range of the sheet metal products,

any improvement in their CAE analysis will have a significant impact on the product development process of all these domains. Hence the domain of sheet metal parts is being considered in the proposed research.

Thus, the present research work focuses on proposing an approach to generate a well connected midsurface, which will improve results of CAE analysis of sheet metal CAD models. The approach is elaborated in details in the following chapters of this thesis.

1.6 Organization of the Thesis

The subsequent organization of the thesis is as follows:

Chapter 2 reviews the relevant literature on the traditional midsurface generation approaches along with auxiliary approaches such as defeaturing and decomposition. Specific issues and limitations of these approaches are highlighted. The chapter concludes by outlining objectives of the present research work.

Chapter 3 presents an overview of the proposed system to generate well-connected midsurface of sheet metal feature based CAD model. It describes major modules involved with respect to their functional capabilities and interrelationships.

Chapter 4 presents at length the algorithms for defeaturing. It initially outlines the phases and then subsequently presents in details the methodology and the algorithms to generate the simplified feature based CAD model.

Chapter 5 presents the algorithms to transform the sheet metal features to a generalized finite set of features. Such generalized feature based model aids in reducing the complexity of the algorithms.

Chapter 6 details out various strategies to generate midcurves, midsurface patches and dedicated algorithms to join them.

Chapter 7 presents a newly proposed methodology for validating midsurface based on topological considerations.

Chapter 8 demonstrates the capabilities of the proposed system for generating quality midsurface with reference to typical model case studies.

Chapter 9 summarizes the research work highlighting the major contributions and outlines direction for the future work.

Chapter 2

Literature Survey

2.1 Introduction

This chapter presents, in details, a review of relevant literature on reported approaches for generation of midsurface. These approaches have been critically examined and compared to highlight their specific advantages and limitations. The chapter concludes by enumerating the objectives of the present research work.

Midsurface is the most widely used surface representation of thin-walled CAD solid model for CAE analysis.

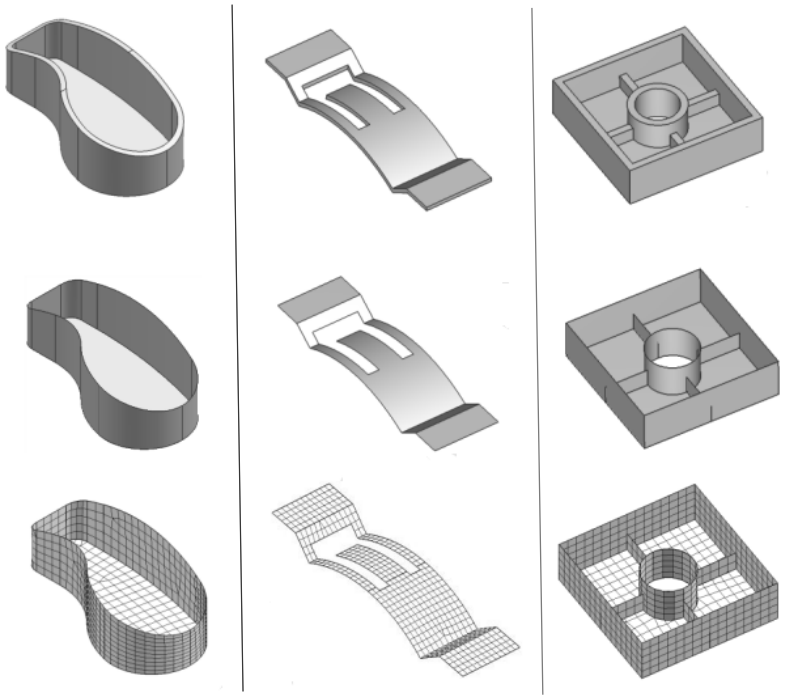
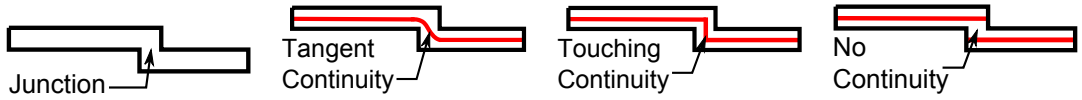


Figure 2.1: Thin-walled Models, Midsurfaces, Meshes (Source: Woo [3])

Figure 2.1 shows 3 thin-walled CAD models in the first row. The second row shows their corresponding midsurface representations. The third row shows shell meshing applied on the midsurfaces for CAE analysis. Midsurface, in different forms, such as me-

dial objects, mid-planes, etc., are being researched for decades. Still, there is no robust, fully automated method for its computation. The reason for this unsolved problem is that there is no single, formal definition of midsurface [8]. Different applications which use midsurface have different expectations about its shape, especially at the connection points. Figure 2.2 shows how midsurface expectation varies [3].



(a) Model (b) Gradual (c) Mimicking (d) Disjoint

Figure 2.2: Midsurface Configurations for a Given Shape

Figure 2.2a shows a step shaped thin-walled solid CAD model represented schematically as a 2D shape. Figure 2.2b shows two midsurface patches being joined in a gradual manner, known as G_1 i.e. tangent geometric continuity. Such midsurface is preferred for CAE analysis. Figure 2.2c shows a planar patch being added to join the two midsurface patches. 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 2.2d shows midsurface patches disconnected and such output can be used where the disconnect needs to be highlighted. A closer observation will reveal that all the output midsurfaces, shown in Figure 2.2, vary in the continuities between midsurface patches at the connections. These continuities are of 3 types: touching (G_0), tangent (G_1) and curvature (G_2). Continuities shown in the in Figure 2.2 are of types $G_1, G_0, 0$ respectively. The present research works aims at producing midsurface with G_0 continuity.

Although there is no single, formal definition of the midsurface, some researchers have specified it semi-formally, as below:

Definition 1 *Mid-surface is an aggregation of surface patches (where each patch corresponds to a pair of non-adjacent surface patches (faces) in the object that are closest to each other) that form a closed and connected set and that satisfy homotopy (Ramanathan [8]).*

Definition 2 *Midsurface is expressed as contiguous flow of the input solid's shape (Rezayat [36]).*

Primary usage of midsurface is in the CAE analysis of thin-walled parts models for placing surface elements such as "shell" elements. These elements are considered superior over solid elements, such as hex (Hexahedral) or tet (Tetrahedral), as they are computationally faster and still deliver fairly accurate analysis results.

Figure 2.3 shows CAE analysis results plots of a simple plate structure. The first figure shows the result with shell elements and the second shows with tet elements.

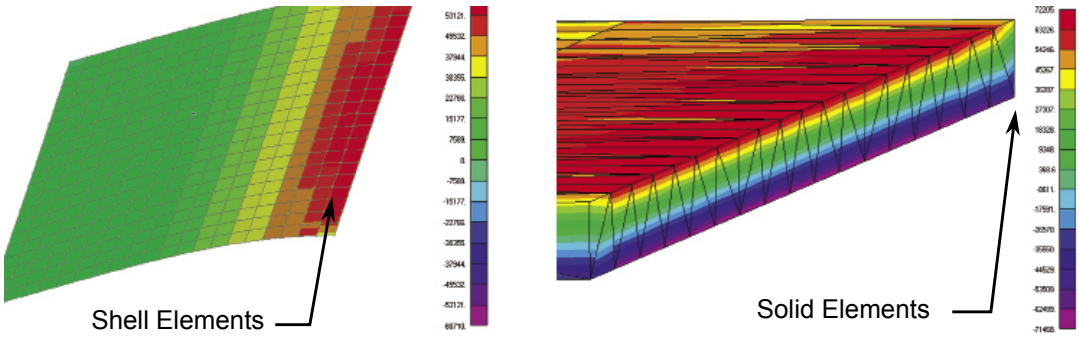


Figure 2.3: Comparison of CAE Analysis with Solid/Shell Elements (Source: Digital Eng [4])

Shell elements give peak bending stress and tip deflection, very close to the theoretical calculations. Tet elements give good deflection values, but poor stress values. The tet mesh is only one element deep. To get accurate stress values, the tet element mesh needs to be at least three elements deep. To avoid the high aspect ratios, in a sample case this may drive the tet element count up towards 1 million, vs. 400 for the shell elements mesh [4]. Thus shell element mesh is far superior in analyzing thin-walled models. A well-connected midsurface is a critical requirement for placing the shell elements.

In some cases, even for relatively thick portions, analysts use midsurface as it gives them the flexibility to play with the thickness which is not possible in case of 3D elements. While solid elements could be used for thin-walled parts, they are inefficient since many more solid elements are required to capture the same response a shell mesh can. At worst, if too few solid elements are used in a bending dominant condition, the results may appear to be correct but could be overly stiff. So, for thin-walled parts shell elements on midsurface are preferred [37].

Although there are many approaches to generate the midsurface, following section details a representative, traditional and one of the most popular approach of generating midsurface.

2.2 Traditional Approach for Generating Midsurface

Midsurface generation is one of the sub-types of dimension reduction process. It, along with defeaturing, forms overall "Model Simplification" process, which is performed during preprocessing of CAD model for CAE analysis. Traditionally, both, defeaturing and dimension reduction, are done together, one after another.

Figure 2.4 shows a representative approach, a schematic outline of overall model simplification process presented by Hamdi [5, 38–41].

CAD model is the input to this approach. It can be either feature based (denoted as "Design by Features" in 2.4) or non-feature based (i.e. Brep model). If it is Brep model, then it undergoes feature recognition process to identify features, thereby mak-

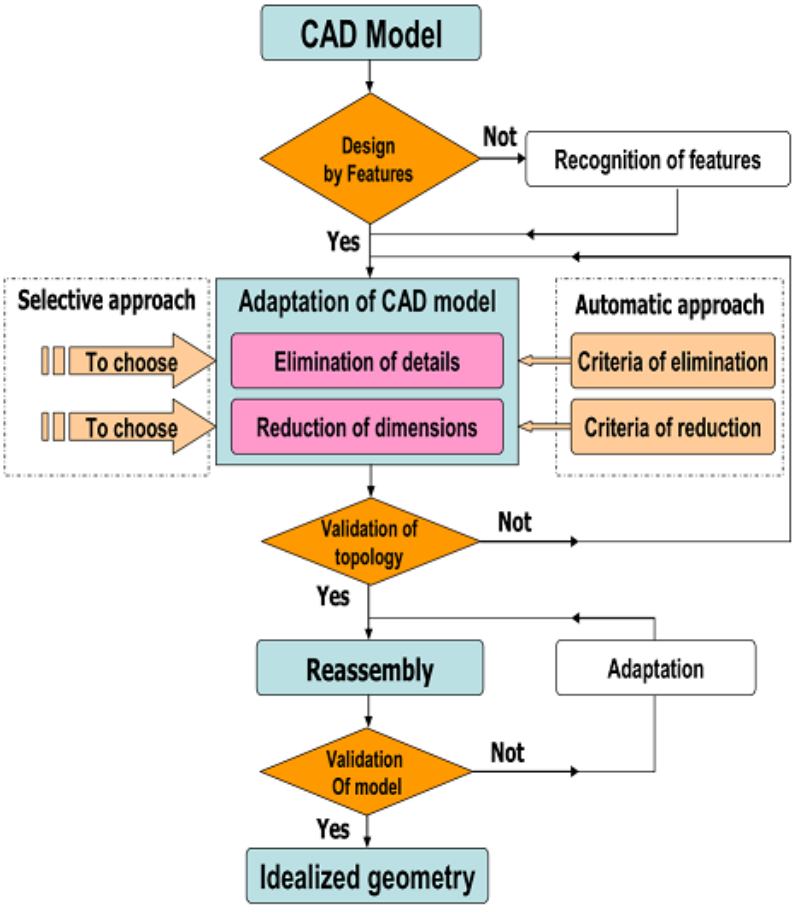


Figure 2.4: Model Simplification Approach (Source: Hamdi [5])

ing it 'feature-based CAD model'. At the "Model Simplification" stage (called "Adaption of CAD model" in Figure 2.4)), both defeaturing (called "Elimination of details") and midsurface generation (called "Reduction of dimensions") are performed together, one after another. In defeaturing, irrelevant features are removed, with 'irrelevance' specified by 'Criteria of elimination'. In dimension-reduction midsurface is generated, with 'thin-walled' portions selected based on 'Criteria of reduction'. The output midsurface is validated and if successful, (shown as as 'Idealized geometry') sent to further downstream applications.

Figure 2.5 shows another traditional approach with more details of actual midsurface generation process. The input CAD model (called "Solid Model") is defeatured in the second stage (called "Simplification"). Here, irrelevant features such as small fillets, chamfers, etc. are removed. Midsurface computation is done using Face pairing approach. The third stage (called "Pair Detection") shows how opposite faces are detected and their face pairs are formed. In the fourth stage, midsurface patches are generated. Patches are then trimmed or extended to meet at common edge. The midsurface patches are then stitched to form the final, well-connected, midsurface model.

Both traditional approaches show the sequential execution of defeaturing and mid-

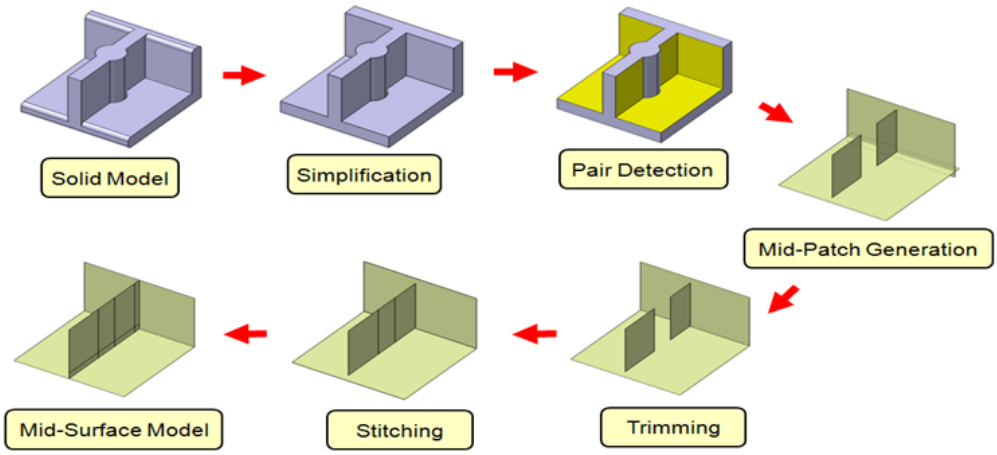


Figure 2.5: Traditional Midsurface Generation Approach (Source: Sheen [6])

surface generation, as part of model simplification process. Finally, the output midsurface is validated for correctness. The following sections detail the state of the art for all of these.

2.3 CAD Model Defeaturing Approaches

Humans, while looking at an object, at the first glance perceive its overall, gross shape and then eventually look into more details as needed [42]. Gross shape is the principal shape that "represents" the given shape, but with far lesser features. In the context of CAD-CAE the gross shape is achieved by defeaturing. Defeaturing is the process of simplification of the shape, by removing small and irrelevant feature details, based on certain predefined objectives or criteria. It is primarily used in CAE analysis where such simplified models lower the complexity of the finite element mesh and thus reduce the analysis time. It is also used in shape matching & retrieval, fast visualization, hiding proprietary details, transmission across the network, etc.

Traditionally, defeaturing has largely been a manual and tedious task. Small and irrelevant features are first recognized in the input mesh or Brep CAD model and are removed manually [43]. Users view this task as too extensive and resort to recreate the necessary geometry than to simplify the existing one [4, 31, 44–46].

Feature-based defeaturing approaches can be classified as:

1. CAD Model Defeaturing based on Feature Information.
2. CAD Model Defeaturing based on Feature Recognition.
3. CAD Model Defeaturing based on Decomposition.

Following section elaborates theses approaches in details.

2.3.1 CAD Model Defeaturing Based on Feature Information

Thakur et al. [47] surveyed and classified various model simplification approaches into four categories, such as surface-based, volume-based, feature based, and dimension-reduction. First three categories are about defeaturing and the fourth is for midsurface generation. From this survey it is observed that most methods were based on the mesh and Brep model as the input, with very few based on the feature based CAD model.

In feature-based CAD, model is built step by step using feature modeling operations. The history of modeling operations is represented in the form of feature tree.

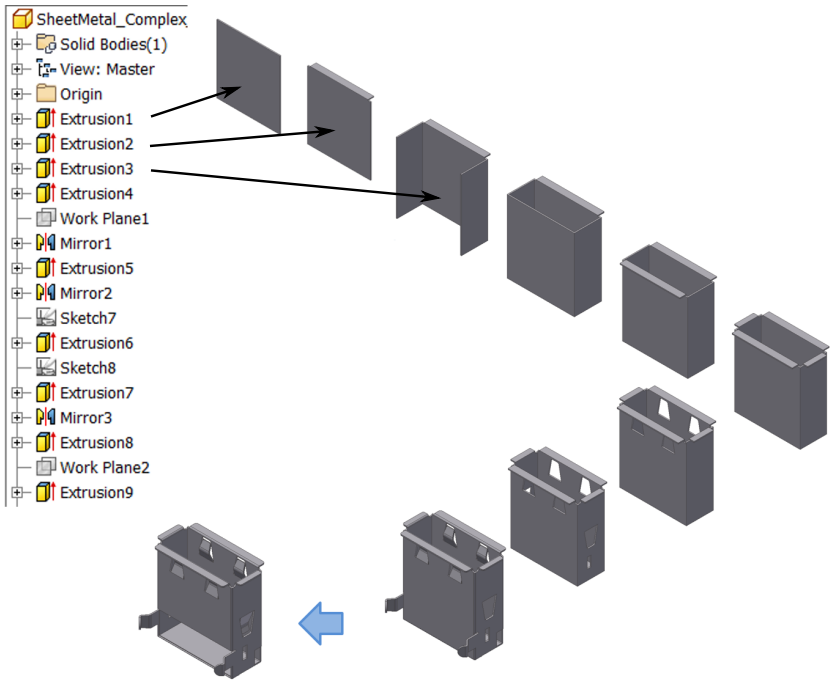


Figure 2.6: Construction of USB CAD Model by Features

Figure 2.6 shows construction of a CAD model of a USB part using a variety of modeling features. At each stage, a new feature gets added to or subtracted from the model built till then. The feature tree is shown at the left. Feature based CAD model, internally builds the model in Brep format as well. So, the final shape, shown at the bottom can be exported in the Brep format. Non-feature based CAD models create Brep format directly and are not built using features. So, features are not accessible from Brep formats.

One of the critical advantage of feature-based CAD model over Brep CAD models is that the individual features can be removed and the model can be regenerated to a valid model without those removed feature. Thus, feature based CAD models present clear advantage over Brep, for defeaturing. Kang [48] observed and reported that feature-based CAD defeaturing operations have better applicability than mesh or Brep model based simplification methods for product design and engineering applications.

Some of the notable feature based CAD defeaturing approaches are reviewed below:

Dabke [49] through the concept of 'global idealization', was one of the first researchers to leverage feature information for defeaturing. His method was based on expert system with heuristic rules derived from the analyst's experience. His approach was however rudimentary in the usage of features.

Smit [50] surveyed various approaches for CAD-CAE integrations and concluded that since features carry domain-specific information, they can bring context-relevant defeaturing. But, the limitation he stated was that many features are built using entities from the existing features, creating dependencies called "parent-child" relationships. Removing the parent feature removes the child features too. Thus, deciding the eligibility of removal of the feature should include similar evaluation of child features also. Otherwise, one has to build or adopt the part in such a way that the dependencies are removed or rerouted first before defeaturing [51].

Hamdi et al. [52] surveyed defeaturing techniques and classified them based on the input format, features simplified, defeaturing criterion, advantages, limitations and application domain. Most of the methods were Brep-based and removed features like holes, chamfers, fillets, protrusions, depressions, passages, concave regions, etc. They used size threshold as well as application-specific rules for identification of the removable features.

Russ [53] mentioned that the determination of the non-critical (suppressible) features relies on different attributes of not only the features themselves, but also of the entire part model and analysis. Some of these attributes include the feature type, feature dimensions, proximity of features to the boundary conditions, analysis type, and part dimensions. He used full feature parameters for deciding the eligibility of the features for removal.

Danglade [54] used machine-learning techniques to capitalize the knowledge and experience of CAE analysts for defeaturing. After a large number of learning trials, the system itself becomes capable of deciding relevance of features. This approach requires feeding of huge initial data to be effective for usage in the real life applications.

Kang et al. [48] customized the defeaturing criteria for shipyard requirements where, apart from geometric reasoning criteria such as volume, they included application-specific rules related to ports and outer boundary of the model.

Thus, for feature based CAD model defeaturing, although having ready access to features reduces the complexity of removal and regeneration of the model, the 'selection of feature for removal' itself remains a challenge. In case of mesh or Brep CAD models, as seen in Figure 2.4, they do not have ready access to features, thus need to undergo feature recognition process to populate the features. Then, these features are evaluated for removal. Following section reviews defeaturing approaches based on Feature Recognition.

2.3.2 CAD Model Defeaturing Based on Feature Recognition

Mesh and Brep are widely used data formats to represent CAD models. In the context of defeaturing, they lack much needed ready access to features. Feature recognition (FR) approaches are used to identify small-irrelevant features in them, which are then evaluated for removal.

Even in case of feature based CAD modeling paradigm, single shape can be modeled with two different feature sequences. So, the feature trees may be different for similar shape. Defeaturing based on features, along with different dependencies, may yield different results. To achieve consistent defeaturing results independent of the modeling history, some attempts use just the final Brep model, and then recognize the features [55] to be removed directly from that final model.

Sandia report [56] recognized features like holes, fillets and chamfers on mesh model and suppressed the smaller ones. In one of the test-cases, the report observed that, the polygon mesh count reduced 10 times. One of the drawback of the approach presented was that it did not incorporate application or domain specific rules for removal of the features.

Belaziz et al. [43] proposed morphological analysis of Brep models based on recognition of form features and using their transformations for simplifications and idealizations. They constructed gross shape step by step by detecting form features.

Joshi Datta [57] recognized features such as holes-fillets-bosses based on type of free form surfaces. Their approach was limited in the features it could recognize.

Hamdi et al. [5, 38–41, 52] eliminated details by merging faces. They also took into account CAE input such as load path BCs. Their method was simplistic in the range of shape handled, for both, defeaturing and idealization.

Woo [55] proposed a method to recognize subtractive features on Brep model, thus eliminating the problem of looking at the history tree. This approach lacked coverage in terms of variety of features being recognized and thus, was limited to simple shapes.

Recognizing features on mesh or Brep CAD model is challenging and thus, it often resorts to heuristic rules, making it error prone. It does not work well in real-life parts having various features which are often interacting with each others. Following subsection evaluates defeaturing by decomposing the input model into primitive shapes and then performing defeaturing.

2.3.3 CAD Model Defeaturing Based on Decomposition

Decomposition is one of the most effective methods for reducing the complexity of the model [58]. Brep, being one the most commonly used modeling format to represent solids in CAD, volume-based methods are used on Brep solid model to decompose it into sub-solids. In volume based decomposition methods, the input solid is partitioned

by cutting planes. Cutting planes are selected based on the application requirement. Certain applications need Convex Partitioning, where cutting planes are placed at concave edges, such that, after partitioning, the sub-solids obtained are of convex type.

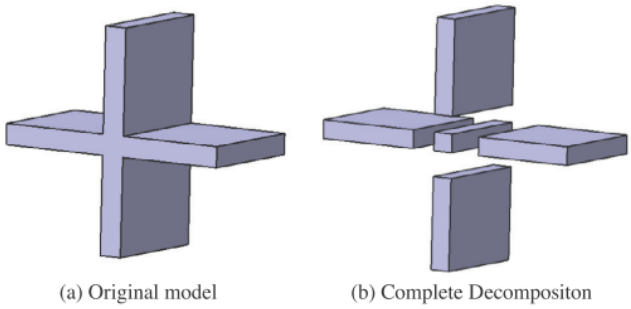


Figure 2.7: Convex Partitioning (Source: Zhu [7])

Figure 2.7 shows convex partitioning of the input solid shape, mentioned as "Original model" into convex sub-solids, shown under "Complete Decomposition". Decomposition has been used effectively for defeaturing. Some of the salient approaches are mentioned below.

Sang Hun Lee [45, 46, 59] elaborated a method to reorder features in the history tree and then to re-execute the history of the reordered features up to the given level of details (LoD). Since the model re-evaluation is computationally intensive, he used cellular topology (CT) created by decomposition, for increasing the performance. One of the major limitations of this approach is that once the model is converted to CT, its feature update capability would cease to exist, making it difficult for any further modifications.

Byung Chul Kim [60] generated feature based model by Cellular or Wrap or Split or Feature decomposition and then suppressed by type and size. This method can be applied widely but was limited in the recognition capabilities.

Decomposition, like feature-recognition, is highly heuristic and error prone, especially for complex models. Thus, defeaturing using decomposition, has limitations while using for real life complex part models.

Table 2.1 presents the summary of the above mentioned defeaturing approaches:

Author	**Input**	**Method**	**Approach**	**Advantages**	**Limitations**
Dabke [49]	Features	Feature size evaluation	Knowledge based idealization	Initial leveraging of feature information	Very primitive Feature recognition

Author	Input	Method	Approach	Advantages	Limitations
Smit [50] [29]	Features	Full feature size evaluation	Less than threshold	Optimized remeshing	Wrong size criterion; Parent-child dependencies
Danglade [54]	Features	Machine Learning	Capturing engineering judgement	Expert knowledge in rules	Large trials
Kang et al. [48]	Features	Importance Rank	Feature rank and application rules	Application customization	Wrong volume criterion
Sandia [51]	Features	Feature size evaluation	Automatic feature routing	Simplified references	Manual
Sandia [56]	Brep	Feature recognition; Size	Recognizes Suppresses small features	Effective simplification	No Application rules
Belaziz [43]	Brep	Feature Recognition	Morphology based simplification	Generic Recognition	Manual selection
Joshi Datta [57]	Brep	Feature Recognition; Feature type based	Sheet metal features Feature Recognition	Holes fillets	Limited primitives
Hamdi et al. [5] [38] [39] [40] [41] [52]	Brep	Feature Recognition; Size based Defeaturing; Proximity; Boundary Conditions	Merging faces	CAE input considered	Orientation not considered
Jae Yeol Lee [61]	Features	Cellular; Size	Cellular; reorder by volume; suppress by size	Efficient Level of Details	Loss of history

Author	Input	Method	Approach	Advantages	Limitations
Sang Hun Lee [59] [46] [45]	Features	Cellular Reordering	Reorder features; Cellular model for performance.	Adjustable Level of Details and Level of Abstraction	Cannot update model
Yoonhwan Woo [55]	Brep	Cellular; Feature recognition; Size	Recognizes and suppresses negative features	Independent of the modelling history	Limited to simple shapes
Byung Chul Kim [60]	Brep	Cellular; Feature recognition; Size	Feature Recognition; Wrap; Split;	Wide applicability	Feature Recognition difficulties

Table 2.1: CAD Model Defeaturing Approaches

2.3.4 Observations on CAD Model Defeaturing Approaches

Research in defeaturing approaches has been going on for decades and remains a challenging problem to solve. Varied requirements from different domains, complexities of input parts and the critical role of engineering judgment in the selection rules, has made defeaturing a challenging task. Following is the list of some of the important observations from the survey of above CAD model defeaturing approaches:

1. Most defeaturing approaches take either Brep or mesh as input CAD model. Small irrelevant features are recognized and removed. Feature recognition, being a heuristic methodology, is not successful on many complex parts. Thus, success of defeaturing is limited and non-deterministic.
2. In case of feature based CAD models, due to ready access to features, removal of irrelevant features and subsequent regeneration of the model is easier. The challenge remains in selection of features to be removed.
3. Size based defeaturing, where features below certain size threshold are removed, is the most prevalent approach. Apart from this, few approaches suggest use of application specific engineering judgment as well, such as, holes in the load path should not be removed however small they are, etc.
4. Full feature dimensions are used by some of the approaches for deciding size

based removal of features. Features volumes, typically are consumed partially in the model when they get added. Thus using full feature dimensions gives wrong candidates for defeaturing.

5. Feature dependencies fetch child features which are not originally selected for removal. These dependencies should be minimized by appropriate modeling practices.
6. Identification of the irrelevant features, only by their feature types, cannot be used blindly across all domains. For example, a rib-like feature may not be relevant in the metal flow analysis, but may be relevant in the heat transfer analysis.

It is thus, observed that CAD model defeaturing approaches largely do not leverage feature information and are not customized to the application domain.

Following section elaborates the second important aspect of the model simplification, i.e. the dimension reduction approaches, specifically the midsurface generation methods, along with their state of the art.

2.4 Midsurface Generation Approaches

Midsurface is part of a family of geometric entities known as "Medial" objects. Other members of the family are Medial Axis Transform, Skeleton, Symmetric Axis, etc. [62]. A medial object represents the input solid shape with one or two dimensions less and lies midway of it [63]. Medial objects such as skeleton and medial surface, are 2 and 1 dimension less than the input solid shape, respectively.

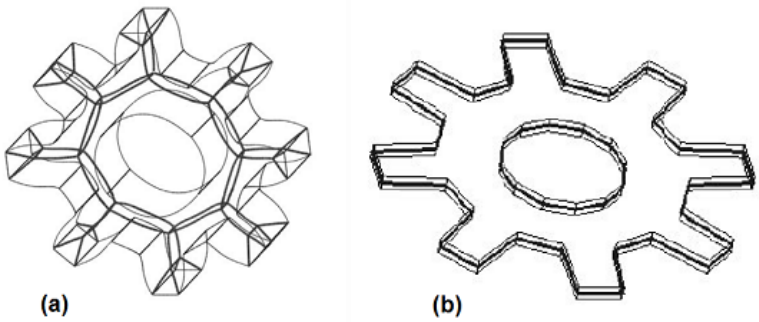


Figure 2.8: Skeleton and Medial Surface of Gear Shape (Source: Ramanathan [8, 9])

Figure 2.8a shows a 1D representation, called "skeleton", of a 3D gear shaped solid. Figure 2.8b shows a 2D representation, called "medial surface", of a similar 3D gear shaped solid.

Medial objects are used for different applications, such as pattern recognition, approximation, similarity estimation, collision detection, animation, matching, CAE, etc. Processing a medial object is quicker as they are simpler, of lower dimensions and still represent the input shape by mimicking it faithfully. Some of the widely used medial

object computation approaches are Medial Axis Transform (MAT) [64], Chordal Axis Transform (CAT) [65], Thinning [66], and Pairing [36], etc.

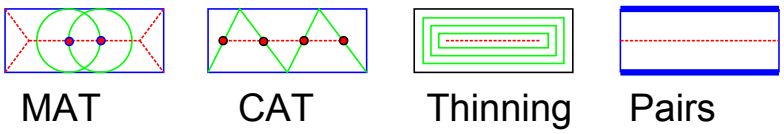


Figure 2.9: Medial Generation Approaches

Figure 2.9 shows their schematic representations. Lines in the middle (dotted) depict medial object of an input shape represented by outer rectangle. In MAT, medial is obtained by tracing loci of the centers of maximal disk, shown as circle, traversing within the input shape. In CAT, the input shape is triangulated first, then medial is obtained by joining midpoints of the sides of the triangle. In Thinning, medial is obtained by successively offsetting the boundary of the input shape, till no further offset is possible. In Pairing, opposite geometric entities are found and their pairs are formed. Medial is computed as an entity equidistant from the paired entities.

Approaches shown in Figure 2.9 can be classified into two categories, viz. formal and heuristics. In formal approaches, a medial object is computed mathematically, in a formal manner, whereas in heuristic approaches, the medial object is computed based on empirical, rules-based procedure. Out of the approaches shown in Figure 2.9, MAT, CAT and Thinning are considered as formal approaches, whereas Pairing is classified as a heuristic approach. Following sub-sections elaborate these approaches and towards end of each category, comparison table is presented along with the summary of observations.

The formal approaches are:

1. Medial Axis Transform (MAT).
2. Chordal Axis Transform (CAT).
3. Thinning and Straight Skeleton.
4. Parametric Equations.

The heuristic approaches are:

1. Face pairing or Midsurface abstraction.
2. Feature-based midsurface.
3. Decomposition based midsurface.

Following section elaborates theses approaches in details.

2.4.1 Generating Midsurface by Medial Axis Transform (MAT)

MAT is a locus of the center of an inscribed disc of maximal diameter as it rolls around the object interior. Figure 2.10 depicts representation of MAT in 2D (called

Medial Axis Curve). MAT algorithms, typically, utilize Voronoi diagrams and Delaunay triangulation.

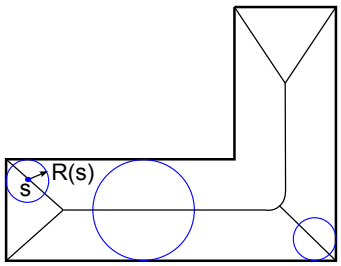


Figure 2.10: Medial Axis Transform (Source: Sheen [10])

MAT was first introduced by Blum [64] in 1967. He proposed Medial Axis Transform (MAT) as a representation that embodies the skeleton of an object as well as the width of the object at every point on the skeleton.

Robinson [67, 68], Stanley [69] used the MAT approach, either to detect thin portions or to compute midcurve.

In one of the recent approaches, Automex [34] used Scale Axis Transform, a modified MAT method, where rolling balls are scaled to address the issue of branches and perturbations. Although the approach is promising, it suffers limitations in complex cases where it creates unnecessary topological fragments.

Ramanathan [8] pointed out that the MAT falls short in its ability to reflect the local topology of the part exactly. This is because of the extraneous portions and non-linear entities that occur due to convex and concave corners in the domain. He modified the MAT method to remove undesired branches. Figure 2.11a depicts the usual MAT output showing branches such as $A-E1, B-E1, F1-C, F1-D$, whereas Figure 2.11b shows midcurve output where spurious branches have been replaced by two extension forming a continuous line, thus mimicking the original shape.

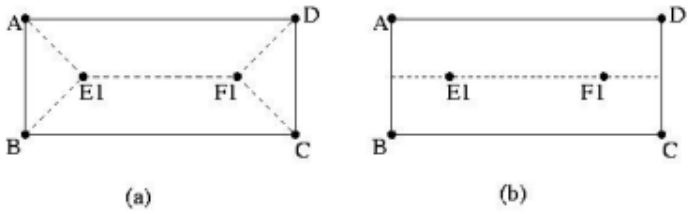
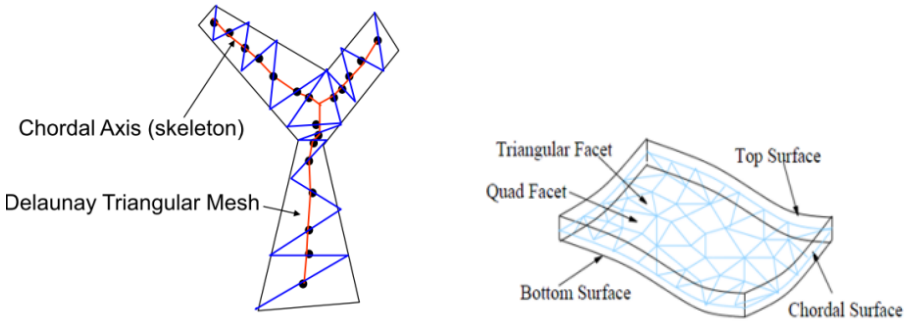


Figure 2.11: Modified MAT for Midcurve Computation (Source: Ramanathan [8])

Many post-processing methods have been proposed to remove branches undesired compared to midsurface expected (Fig. 2.9) [70–78], but they are not practically useful.

2.4.2 Generating Midsurface by Chordal Axis Transform (CAT)

For computing CAT, in case of a 2D shape, the input is meshed by Constrained Delaunay Triangulation (CDT). Midpoints of the chords i.e. sides of the triangles, are joined to form the medial [11]. In case of 3D solid shape as an input, a tetrahedral (tet) mesh is generated without inserting interior nodes and then midsurface is generated by fitting through midpoints of tetrahedral elements' faces, called chordal faces. Figure 2.12 shows CAT approach applied to both, 2D and 3D input shapes. Figure 2.12a shows how 2D input shape is meshed with triangles and then a curve is fitted through the midpoints of the triangle edges (chords). Figure 2.12b shows result of tetrahedral elements meshing and computation of CAT by fitting a surface through midpoints of the tetrahedral elements faces (chordal faces).



(a) CAT Curve for 2D Input (b) CAT Surface for 3D Input

Figure 2.12: Chordal Axis Transform (Source: Quadros [11])

CAT was first proposed by Prasad [65] and developed extensively by Quadros [11, 79, 80]. Quadros [79–81] triangulated the input shape with single layered constrained meshing, classified cells based on positions and had each unit compute their own midsurface patches, which later were joined. With this approach, medials of large, complex parts can be calculated, but the limitation is that getting the uniform, one layered triangulation itself is challenging. Thus, this method is not used for practical CAE analysis.

2.4.3 Generating Midsurface by Thinning Approaches

Thinning approaches simulate a grass-fire like process [64]. In the grass-fire process, when boundary of grass field is set to fire, the fire front successively shrinks the field inwards till it reaches the intersections. In the context of medial computation, the grass-fire process is simulated by making object thinner and thinner in successive iterations by offsetting its boundary toward its interior until it vanishes.

By iteratively offsetting the boundary curves and efficiently identifying the breakpoints, Montanari [82] could locate all the critical points on the medial axis, together with the geometry of the skeleton segments that connect them. Gursoy [83] extended his work by including circular arcs as boundary elements.

Aichholzer [12, 84] was one of the first proponents of a special thinning method called “Straight Skeletons”.

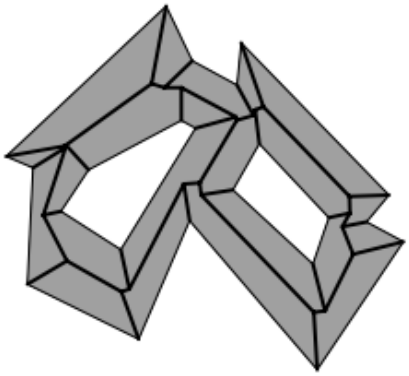


Figure 2.13: Straight Skeletons (Source: Aichholzer [12])

Figure 2.13 shows an example of Straight Skeleton of a 2D shape. At each vertex, angle bisectors are produced, which are used to partition the interiors. The lines joining intersection pins of the bisector forms the medial curve. Although this method can take both, thick and thin shapes, it is restricted to only polygons. Another limitation is that, similar to MAT, it also produces unwanted branches which need to be removed.

2.4.4 Generating Midsurface by Parametric Equations Approach

Parametric approach takes two parent curves and fits a curve through equidistant points from both the curves [13, 85]. Figure 2.14 shows two input curves $C_1(u)$ and $C_2(v(u))$ generating midpoint $O(u.v(u0))$. Loci of the Os is the midcurve. The same approach is extended to surfaces as well.

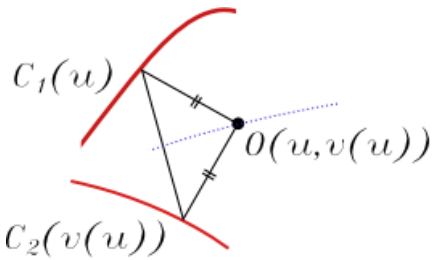


Figure 2.14: Parametric Midcurve (Source: Fischer [13])

In Fischer’s [13] parametric approach, sketch profile of features like Extrude, Revolve is extracted. The piecewise linear midpoints of intersecting pair of matched normals are generated. They are then fitted with the Bspline midcurve. Midcurve is extruded or revolved to generate the midsurface. This approach gives fairly good results in generating midsurface patches per feature, however, does not elaborate on joining connections between the midsurface patches.

In one of the most recent approaches reported by Zhu et. al [7, 58], the input surfaces

are triangulated, then for each point pair a middle point is computed, which are then fitted with a surface, resulting into a midsurface. While this method removes need for post-processing self-intersecting surfaces, it has major drawbacks that it needs surface pairs to be readily available to begin with and requires expensive triangulation.

Table 2.2 outlines the summary of relevant formal approaches.

Author	**Input**	**Medial**	**Approach**	**Advantages**	**Limitations**
Harry Blum [64]	Curves	MAT	Grass Fire	Any input shape	Branches
Ramanathan [8]	Curves	Midcurve	Removed erroneous MAT branches	No branches	Limited shapes
Robinson [67] [68]	Brep; Sketch	MAT	Rolling Ball	Thinness detection	Post processing
Stanley [69]	Curves	MAT	Voronoi Diagram	Any input shape	Branches
Automex [34]	Brep	Scale Axis Transform	Scaled spheres to remove branches	No perturbations	Incorrect Fragments
Prasad [65]	2D 3D shapes	CDT	Mesh the shape; join midpoints of the triangles	Any shape can be the input	Dependence on triangulation; gaps at the end
Quadros [81] [11] [79] [80] [11]	Brep	CDT	Constrained meshing; cellular midsurface	Handling Complex parts	Triangulation problems
Montanari [82]	polygon	Offset	Iterative offsetting; breakpoint identification	Any input shape	Only polygons ; extra branches
Gursoy [83]	Curves	Offset	Offsetting circular arcs	Curved shapes	Branches
Aichholzer [12][84]	polygon	Straight Skeleton	Angular bisectors as partitions	Any input shape	Only polygons ; extra branches

Author	Input	Medial	Approach	Advantages	Limitations
Fischer [13]	Sketch Profile	Normal Bisector	Midpoints at normal intersection; B-spline fitting	No branches	Post processing at intersecting midcurves
Zhu [58] [7]	Surfaces	Normal Bisector	Midpoints at normal intersection; surface fitting	Any surface	Sampling errors

Table 2.2: Formal Midsurface Generation Approaches

2.4.5 Observations on Formal Midsurface Generation Approaches

Approaches based on MAT, CAT, Thinning and Parametrization have been in research for decades. All of them, being formal-mathematical, can be applied to a wide variety of shapes, and not restricting themselves to only thin-walled shapes. Following are some of the salient observations with regards to these approaches:

- Biggest strength of the formal approaches like MAT is that it can be computed for any shape, thick or thin. Being formally defined, the converse or reversal process (meaning "given a MAT, compute the original shape") is possible.
- Major drawback of MAT, Thinning methods is that it creates unnecessary branches and its shape is smaller than the original corresponding faces.
- MAT based approaches also suffer from robustness problem. A slight change in base geometry forces re-computation of MAT and the results could very well be different from the original.
- Although MAT approaches have been around for decades and are fairly mature, its usage in midcurve-midsurface generation is still very complex and difficult to ensure appropriate topology.
- The major limitation of CAT approach is that mesh has to be generated before. Creating constrained, single layer meshes on complicated CAD models are, at times, difficult.
- Thinning approaches are based on split events of the straight line skeleton giving counter-intuitive results if the polygon contains sharp reflex vertices.
- In Parametric approach, the two input curves or surfaces may not be in one-to-one form. In such cases maintaining continuity can be challenging.
- Quality of surface generated by parametric approach depends on the sampling done to compute the midpoints.

Thus formal approaches to compute midcurve or midsurface have some major limitations, rendering them less useful for the practical scenarios. Following approaches are more heuristic in nature and they are found to have wider applicability in the practical scenarios.

2.4.6 Generating Midsurface by Face Pairing Approach

Midsurface Abstraction (MA) or Face-pairing method was first proposed by Rezayat [36]. In this approach, face pairs are identified on the thin-wall part's Brep model. It involves casting a ray from a face in the Brep's material direction, to find the opposite face. Such opposite faces form face-pairs. Each face pair generates its own midsurface patch. Connectivity information between respective face-pairs is used to decide which midsurface patches join each other. The midsurface patches are joined, either by trimming or extending them to come to a common location.

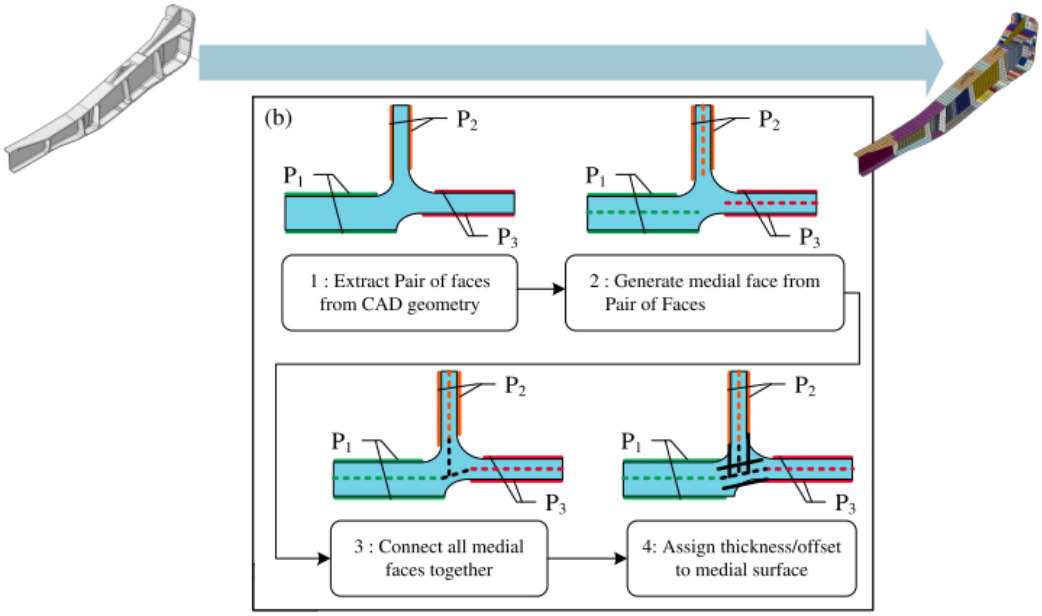


Figure 2.15: Generating Midsurface by Face Pairing (Source: Boussuge [14])

Figure 2.15 depicts the face pairing approach. Figure 2.15:1 shows face pairs being detected. In Figure 2.15:2, midsurface patches are generated for each face pair. In Figure 2.15:3, patches are extended or trimmed to join at common locations. In Figure 2.15:4, thickness values are assigned on the midsurface.

Rezayat claims that the face-pairing approach has benefits over the MAT approach because the resultant geometry is cleaner and requires less post-processing. However the face pairing approach suffers from severe difficulties especially in complex models, such as identifying correct face pairs and joining midsurface patches properly.

S H Lee and D P Sheen [10] proposed Solid Deflection method leveraging face-pair detection and face geometry replacement. Topology was kept intact while bigger

of the two faces was offset-ed. Tolerance was increased so as to make them look like connected but they were not. Their work appears to be very limited in the range of input shapes it would handle.

Sungchan Kim et al. [86] developed various approaches for model simplification, mainly focusing on defeaturing aspects, but they also mentioned usage of face-pairing for dimension reduction in case of thin solids. Their approach was simplistic and lacked details on joining of midsurface patches.

Face-pairing algorithms are implemented in the commercial CAD-CAE applications. They have automatic as well as manual input. Thus, despite the limitations, these applications are able to generate mid-surfaces for a range of realistic models. Approaches seen so far were based on Brep model. Following section elaborates midsurface generation approaches using feature information.

2.4.7 Generating Midsurface by Feature-based Approaches

In feature based CAD, a model is built step-by-step using features. At each step, input parameters of the features are used to build tool-bodies. These tool bodies are then boolean-ed with the model built till that step.

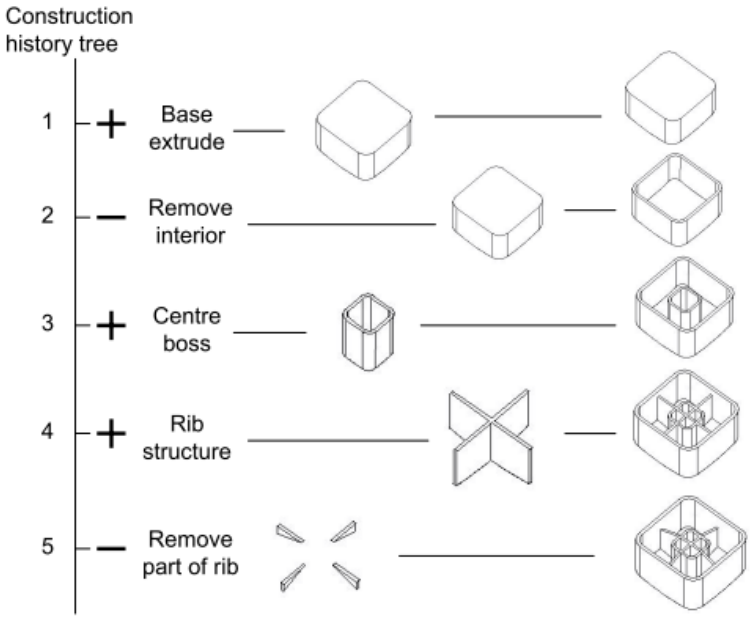


Figure 2.16: Construction of CAD Model by Features (Source: Stolt [15])

Figure 2.16 shows Stolt's [15] depiction of construction history, i.e. feature tree of a CAD model. The shapes shown in the middle are the tool bodies, which are boolean-ed (shown as $+$ and $-$) to build the model. In many cases, the tool-bodies are of relatively simple shapes such as extruded or revolved sections, thus making feature based algorithms simpler and deterministic. There have been a few attempts to use the feature information in the computation of the midsurface. Use of features has not

been done extensively in the past because feature specific information was not exposed by the CAD systems, either through file format or through Application Programming Interfaces (APIs). These days, however, most of the widely used CAD modelers provide APIs to expose feature information, giving rise to their usage in CAD algorithms.

Stolt [87] worked on features such as “Pad”, “Pocket”, extracted sketches and analyzed if there are parallel curves to form mid-segments. These segments were used for extrusion with same parameters as the original feature.

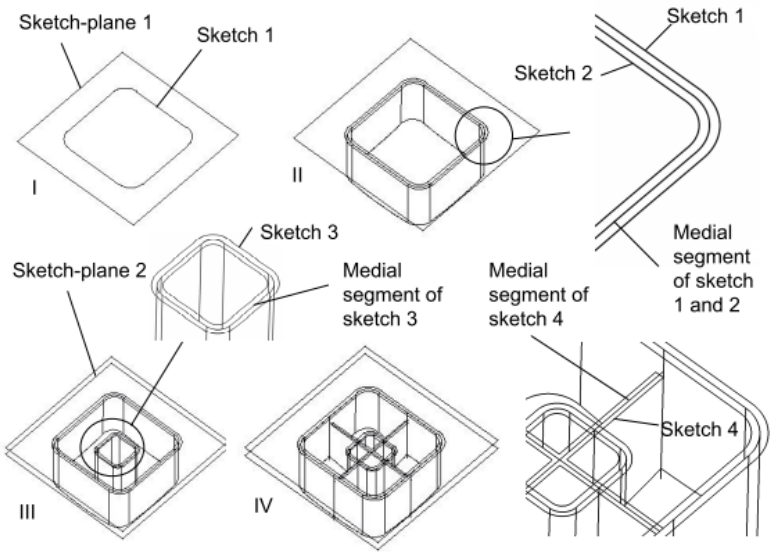


Figure 2.17: Construction of Feature-based Midsurface (Source: Stolt [15])

Figure 2.17 shows the process of building feature based midsurface proposed by Stolt [15]. Figure 2.17:I shows the first feature’s sketch. Figure 2.17:II shows the first feature, along with its midsurface computed from the medial segment of the first sketch. Similarly Figure 2.17III and IV show, features being added along with their midsurface.

Robinson et al. [68] while working on mixed-dimensional model for aerospace structures utilized information in CAD feature tree to locate sketches and then features built on top of it like Sweep, Revolve. The work appears to be limited due to processing of a small set of singular features only and no consideration seems to be given to joining process.

These approaches are comparatively less in number and restrict themselves to midsurface patches creation. They do not leverage feature information effectively in setting up appropriate connections between patches.

2.4.8 Generating Midsurface by Decomposition Based Approaches

Decomposition is a process of partitioning given shape into non overlapping subshapes. It can be applied on both, 2D and 3D CAD shapes. Decomposed sub-shapes compute the medial object. If 2D sketch profiles is the input then, midcurves are com-

puted and if 3D Brep solid is the input then midsurface is computed.

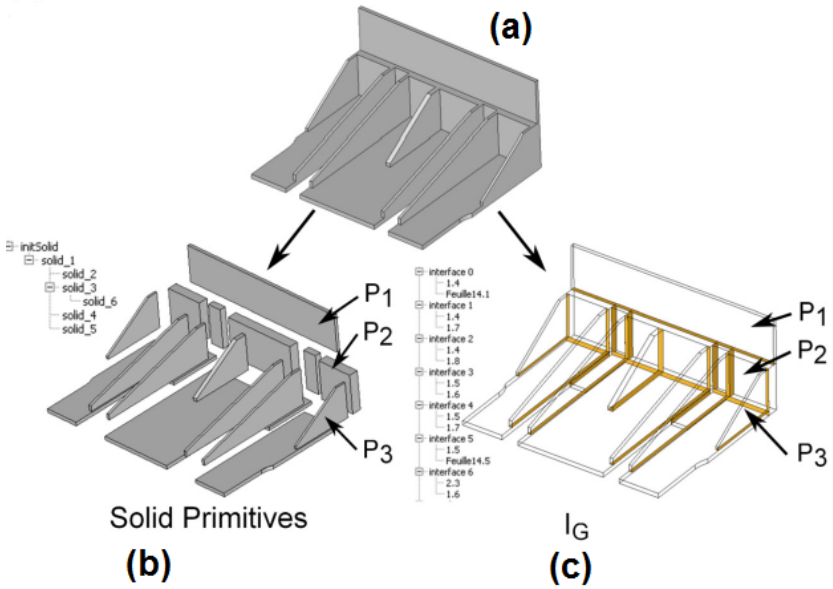


Figure 2.18: Generating Midsurface by Decomposition (Source: Boussuge [16])

Figure 2.18 depicts Boussuge's [16] approach of midsurface by decomposition. Figure 2.18a shows the input Brep solid CAD model. Figure 2.18b shows its decomposition into sub-shapes, called "solid primitives". Figure 2.18c shows midsurface computed for each solid primitive.

The sub-shapes generated by decomposition are called as "Cells". Cells not only simplify the geometries to work with but also introduce possibility of parallelization. With these advantages, researchers have used decomposition extensively in CAD (as voxels), CAM (machining pockets), CAE (meshing), etc.

Quite a few attempts to compute the midsurface using cellular decomposition are found in the literature [3, 88–90].

Chong et al. [88] split the Brep model into sub-volumes which, in turn, extracted their own midsurface patches.

Kageura [17] filed US Patent 2009/0271156A1 in which, as shown in Figure 2.19, an input shape is partitioned first, adjacency information is generated, midsurface patches are extracted and combined.

Cao [89, 90] decomposed the final Brep model and recognized additive remnant swept primitives, got their sketches, computed midcurves for edge-pairs and created midsurface patches but the joining logic was not much elaborated.

Boussuge et al. [16, 91] split the Brep model at concave edges, recognized (only) positive protrusion features in the sub-volumes and then idealized each to a midsurface with a joining logic restricted to only a couple of hard-coded interaction types. Figure 2.18 shows how a model is split into solid primitives, midsurface patches are generated and they are joined at the interfaces I_Gs.

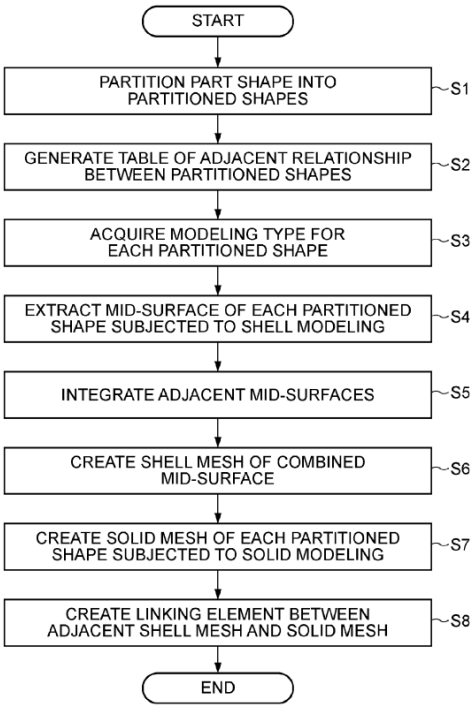


Figure 2.19: Generating Midsurface by Partition Approach (Source : Kageura [17])

Woo [3] used his maximal volume technique [55, 92] and computed midsurface for each sub-volume using face-pair technique. It worked only for analytical surfaces and parallel face pairs

Table 2.3 presents a summary of relevant heuristic midsurface generation approaches:

Author	**Input**	**Medial**	**Approach**	**Advantages**	**Limitations**
Rezayat [36]	Brep	Face Pairing	Face pairing; Midsurface patches; Adjacency Graph; Extend and trim	No branches	Complex Face pairing
Sungchan Kim [86]	Brep	Face Pairing	Defeaturing; Face pair method	Usage of Face pairing on feature-basis	Connection details missing
Sheen [6] [10] [93]	Features	Defeaturing; Face pairing	Feature based model simplification; Face pairing	Better simplification	Co-planar; T and L type only

Author	Input	Medial	Approach	Advantages	Limitations
Aimin [94]	Features	Defeaturing; Face pairing	Defeaturing; Face pair method	Defeaturing parameters collected upfront	Connection details missing
Dabke [49]	Features	Pre Defined	Per feature pre defined midsurface	Feature based CAD-CAE migration	Primitives only
Fischer [85]	Features	Parametric	Parametric midcurves swept	3D midsurface to 2D midcurve	Specific input surfaces only
Stolt [87] [95]	Features	Pre Defined	Midcurves of sections lofted or pre defined from database	Customized to Injection Moulding	Sectioning not perfect
Hamdi [5] [38] [39] [40] [41] [52]	Features	Pre Defined	Hardcoded pre defined primitive midsurface	Leveraging feature information	Very primitive features only
Smit [50] [29]	Features	Pre Defined	Primitive feature based pre defined midsurface	Design Analysis feature mapping	Connection logic unclear
Aimin [94]	Features	MA	Feature based model simplification.	Defeaturing parameters are collected upfront	No details on how midsurface are connected
Chong [88]	Brep	Concave Edge Partitioning	Edge-pair detection and collapsing	Mixed dimensional modelling	Heuristic-Hardcoded; Analytical surfaces only
Kageura [17]	Brep	Cellular	Cell based Midsurface and joining	Adjacency graph for generalness	Only simple academic shapes

Author	Input	Medial	Approach	Advantages	Limitations
Cao [89] [90]	Brep	Cellular; Face Pairing	Protrusion detection; Midcurves of sketches	Better CAD-FEA integration	Only additive swept features
Yoonhwan Woo [3]	Brep	Cellular; MA	Maximal sub volumes; Midsurface Abstraction per cell	Detailed face pairing	Heavy use of Booleans
Boussuge [96] [91] [16] [14]	Brep	Cellular; Feature Recognition; MAT	Construction history by Cellular. Cellular FR for Extrudes	History tree independent	Only Extrudes. Parallel and Orthogonal connections only.
Huawei Zhu [58]	Brep	Cellular; Parametric Midsurface	Virtual Cellular by Face Group method; classification of connections	Booleans are avoided	Cellular not possible for complex shapes

Table 2.3: Heuristic Midsurface Generation Approaches

2.4.9 Observations on Heuristic Midsurface Generation Approaches

Extensive research has been done on the heuristic methods for computing midsurface, such as midsurface abstraction (face pairing), feature based, cellular decomposition based, etc. These are more widely used in commercial applications, compared to the formal approaches such as MAT, CAT, etc. In-spite of a huge demand, the heuristic methods have limited usage, especially in case of complex models. Following are the salient observations on heuristic methods:

- Face pairing is widely used approach amongst the commercial CAD-CAE applications, as it provides the user the ability to pick appropriate face pairs in case the automatic detection fails.
- Automatic Face pair detection is error prone for complex CAD models as in-

put. Joining midsurface patches also does not work robustly for them. Due to these problems, Face pairing suffers from limitations such as extensive use of hard-coded rules for detecting edge or face pairs, support for limited types of geometries, limited range of connections, etc.

- Midsurface by cellular decomposition approach is not used widely. Cellular decomposition can result in large number of cells, needing quite a few boolean operations, making it ineffective and unstable. Choosing correct splitting logic, maximal volume rules should help make the process optimized and robust.
- Feature based midsurface computation approaches bring definitiveness in computation of midsurface, due to lesser use of heuristics.
- Feature information has not been extensively used for computation of midsurface due to its inaccessibility so far.
- Most current feature based midsurface computation approaches restrict themselves to midsurface patches creation and do not leverage feature boolean information in setting up appropriate connections between patches.

Overall, both formal and heuristic approaches present strengths and limitations in different aspects of computing midsurface. Out of them, use of defeaturing, leveraging feature information and decomposition appear promising in computation of a quality midsurface.

The quality of midsurface is validated by various techniques, primarily to detect failures such as gaps, overlaps, etc. Following section elaborates state of the art for techniques used in validating the quality of the output midsurface.

2.5 Validation Approaches for Generated Midsurfaces

Even after huge demand, extensive research and wide availability in the commercial CAD-CAE applications, quality of the midsurface computed is still a concern. Validating the output midsurface is therefore a critical step in assessing the quality, after which corrective actions can be taken.

Midsurface is expected to express the contiguous flow of the input solid's shape [36]. So, to be truly effective, the output midsurface needs to mimic the input thin-walled solid model, in both, geometrical and topological sense. Geometrically, the shape of the midsurface should be such that it lies in the middle (at half the thickness) of the thin-walled solid model. Topologically, the connectivity between the midsurface patches should be similar to that of their corresponding face pairs in the input thin-walled solid model.

Thus, there are primarily two approaches to validate the output midsurfaces, namely "geometric" and "topological" [27] as explained below:

- **Geometric Validation of Midsurface**: Validates if the output midsurface lies midway of the input solid model. Lockett [27] proposed geometric similarity index based on Hausdorff distance which provides a measure of the maximum dissimilarity between the midsurface and its associated input solid model.
- **Topological Validation of Midsurface**: Validates if the output midsurface has similar topological connectivity as present in the input solid model. Lockett [27] predicted topological entities of midsurface based on the topological entities of the input solid model and then used topological variants for checking the validity of the midsurface. The main limitation of their approach is the use of geometric criteria, such as closest distance proximity or angle between faces, in the topological validations, which is contrary to the concept of topology.

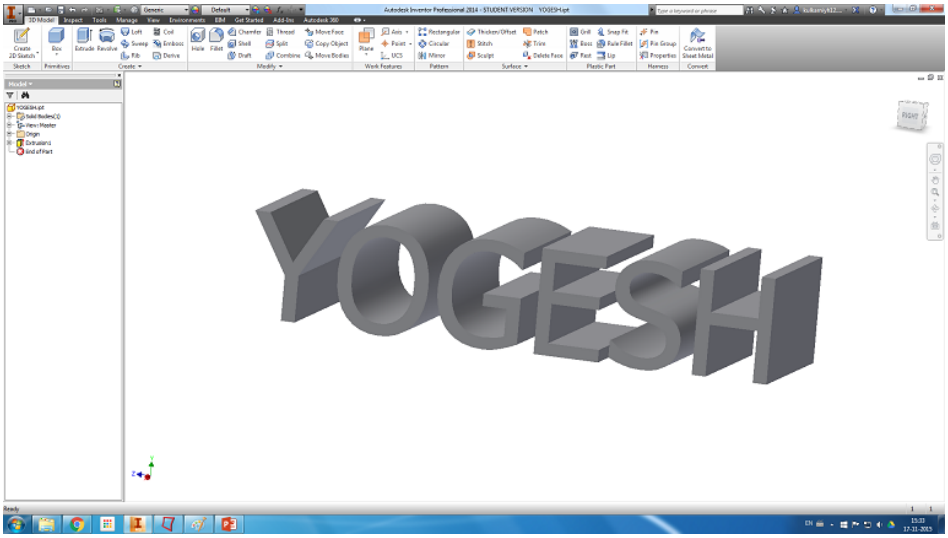
Although geometric approaches of validating the output midsurface are widely used, they are tedious to execute. Topological approaches are not very common and are not yet fully developed to be used practically.

2.6 Midsurface Generation Capabilities of Commercial CAD-CAE Systems

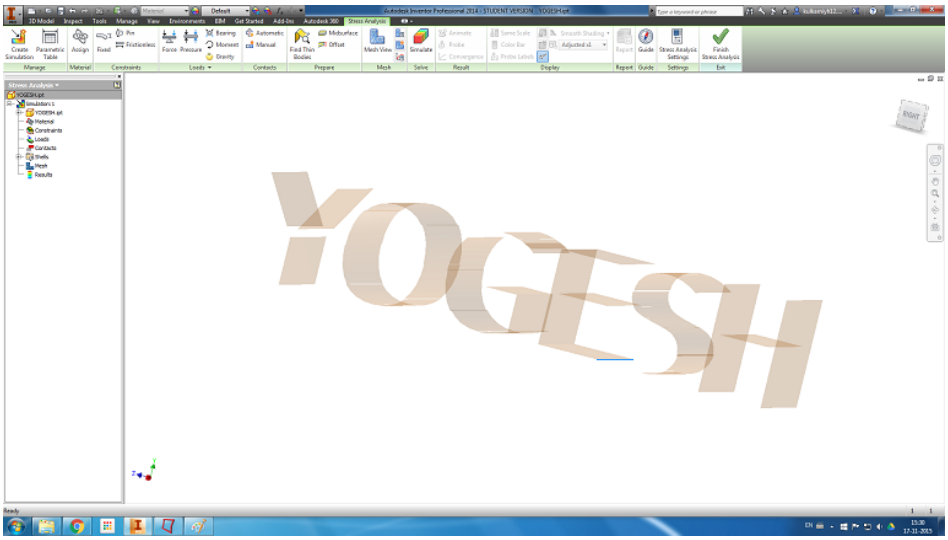
Commercial CAD-CAE systems are in use for decades. Both systems got developed separately and are like islands of automation. The corresponding commercial systems have to transfer model data amongst themselves only through neutral file formats. Thats the reason, most of the CAD-CAE approaches are based on Brep model populated by neutral file formats, such as STEP and IGES. Data interoperability through neutral formats is susceptible to data loss. Thus most of the data coming to preprocessing from CAD systems, has to be cleaned and healed first. But, now a days, with better CAD-CAE systems integrations, native data and in some cases even feature information is made available to CAE systems. With richer information, some of the CAD-CAE systems have improved preprocessing functionalities like defeaturing and midsurface generation.

CAD kernels like ACIS®, modelers like Autodesk Fusion® and CAE packages like Altair's Hypermesh®, Ansys®, MSC Nastran®, etc. do provide defeaturing capabilities mainly for CAE analysis that are primarily size-based in nature.

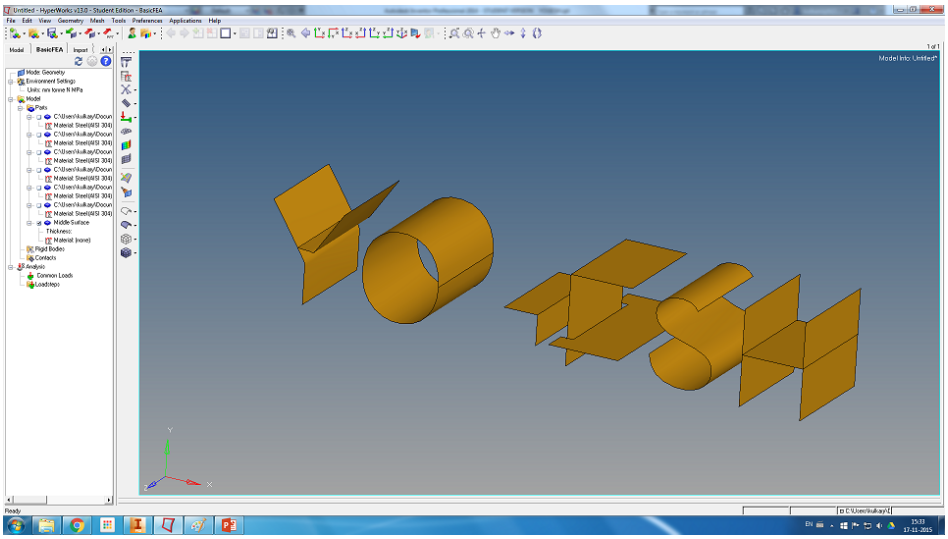
Midsurface functionality is widely available in CAD-CAE systems like Autodesk Inventor®, Parametric's Creo, Altair's Hypermesh®, Ansys®, MSC Nastran®, etc. It is observed that quality of midsurface in CAD systems is relatively lower than the CAE systems. But overall, automatic midsurface generation capabilities do not seem to be robust and meaningful for practical usages, especially for complex parts. Manual intervention and post-processing is used to correct the problems of the output midsurface.



(a) Input Thin-walled Solid Model



(b) Output Midsurface of a CAD System



(c) Output Midsurface of a CAE System

Figure 2.20: Midsurfaces Generated by Commercial CAD and CAE Systems

Figure 2.20a shows the input CAD solid model, having English alphabets shapes. These shapes are chosen for benchmarking, as it is easier to imagine their corresponding midsurfaces. The resultant midsurface is expected to look like those alphabets but in a surface form. Figure 2.20b shows midsurface output generated by a popular CAD system, whereas Figure 2.20c shows midsurface output by a widely used CAE system. Both CAD and CAE midsurface outputs show many errors, such as missing surfaces, gaps, not-lying-midway, etc.

From the above results it is evident that there is still an ample scope for improvement in devising a robust and automated method for computing a quality midsurface.

Another detailed survey was done by conducting technical discussions with CAD-CAE experts, academicians related to various aspects of midsurface generation. This was essentially aimed at understanding industry practices, issues etc. Findings from theses discussions are provided in Appendix A.

2.7 Observations from Literature Survey

After analyzing various reported approaches for midsurface generation, it is observed that there has been limited success in the generation of a quality midsurface. Midsurface generation approaches are computationally intensive and the errors take enormous manual interventions to correct, hence the overall process can take days or even weeks to finish.

Following are some of the salient observations based on the literature survey done:

- In feature based defeaturing approaches, full feature dimensions were used for selecting features for removal thereby giving wrong results.
- For defeaturing, application domain-specific rules of identification of features for removal are necessary instead of identification of any feature below the 'threshold' size.
- Past midsurface generation approaches are limited in the range of input model geometries (say, only planar or analytic surfaces), feature types (say, only extruded, positive primitives) and connection types (say, only, parallel, or perpendicular) they could handle. Due to these limitations, the output midsurface has many errors, especially for complex input solid models.
- In face pairing approaches, finding the face-pairs in a complex part itself is very challenging. Issues like, faces may not be directly opposite to each other, may not be parallel, may have multiple opposite faces, etc. are frequent. Such wrong face pairs result in gaps in the output midsurface.
- For connecting midsurface patches, there is no definitive list of connection types. Thus providing deterministic logic is not possible. This limitation results in mid-

surface patches not being joined properly, resulting in gaps or overlaps.

2.8 Research Objectives

The overall objective of the proposed research is to develop a system to generate a quality midsurface of sheet metal feature based CAD model by developing various algorithms. Based on the extensive literature survey and critical analysis, following research objective are laid down, as:

- To develop defeaturing algorithms to simplify CAD feature model as much as possible without impacting the gross shape of the part.
- To develop algorithms to represent the feature based CAD model by transforming existing set of sheet metal features to a finite set of generalized features.
- To develop algorithms for generating quality midsurfaces leveraging generalized feature based CAD model and cellular decomposition covering a wide variety of topological connectivities.
- To come up with an approach to topologically validate the generated midsurface relative to its input CAD model.
- To implement a software system incorporating above algorithms and demonstrate the efficacy of the proposed approach.

2.9 Research Methodology

The present research work uses empirical hypothesis testing research methodology.

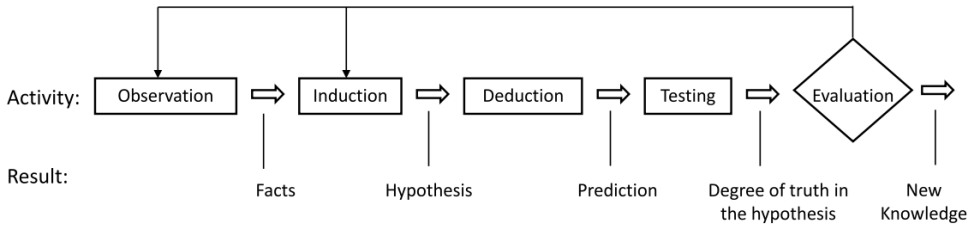


Figure 2.21: Empirical Hypothesis Testing Process (Source: Stolt [1])

Figure 2.21 shows the empirical hypothesis process and its various phases. Following paragraph explains them along with their interpretation in the context of present research work.

The empirical hypothesis testing process starts with observations, e.g. the problems observed in the current midsurface generation approaches. These facts are generalized by induction into a hypothesis, such as, the core problems in midsurface extraction are detection of sub-shapes and interactions amongst them. Predictions are then deduced from the hypothesis, such as, if the input model is simplified, generalized and

decomposed, the problem could be more deterministic to handle. A prototype is built incorporating the proposed solution. After testing the prototype, if the results are valid and acceptable, then the hypothesis is accepted as true for the time being. In the present research work, hypothesis takes the form of a set of multiple sub-hypotheses and the proposed system answers those questions. Section 2.11 lists the hypothesis whereas Chapter 3 presents the proposed system.

2.10 Research Scope

The proposed research aims at developing algorithms for generating a quality midsurface of thin-walled sheet metal feature based CAD model. Reason for considering sheet metal parts in scope is due to its wide usage in domains ranging from household items to big ships. Better midsurface will have an enormous impact on quicker and robust product development in all these domains.

The proposed system accepts feature based CAD model built using Autodesk Inventor®, as input. Thus, the feature based alogirthms devised in the present research, work on sheet metal features, such as, Walls, Flanges, Bends, Cutouts, etc. Autodesk Inventor® was chosen over other CAD systems, such as Solidworks, Parametric Creo, etc. due to its free availability for students and maturity of APIs.

Following section lists hypotheses tested by the present research work.

2.11 Research Hypotheses

This research proposes following hypotheses and assesses them against the proposed system.

Hypothesis 1 *Instead of working on a Brep CAD model as input, the feature based CAD model enables computation of a well-connected midsurface.*

Hypothesis 2 *Instead of working on a detailed CAD model, the defeatured model results in a better midsurface.*

Hypothesis 3 *Instead of working on a wide variety of features, the generalized feature representation helps devise a generic midsurface patch computation algorithm, thus avoiding problems of face pairing.*

Hypothesis 4 *Instead of working on a full CAD model, the decomposed model helps devise a generic midsurface patch joining algorithms, thus avoiding problems of gaps and overlaps in the midsurface.*

Hypothesis 5 *Feature based defeaturing, generalization and decomposition, together help compute a well-connected midsurface.*

This chapter has presented review of the past approaches for generating midsurface along with the associated techniques of defeaturing. It summarized the observations and presented the objectives for the present research. The following chapter will present the proposed system to address the research objectives.

Chapter 3

MidAS - A System for Generating Midsurface for Sheet Metal Feature-based CAD Model

3.1 Introduction

This chapter presents an overview of the proposed system called, **MidAS** (**Mid**surface **A**lgorithms for **S**heet-metal-parts), for generating a quality midsurface, that is designed and implemented in the present research work. It will assess hypotheses mentioned in Section 2.11. Modules in **MidAS** are connected in tandem and comprise of set of algorithms to carry out designated processing. The modules interact with the sheet metal feature based CAD system for exchanging and processing Brep and feature information of the CAD model. Following sections elaborate each module and their functioning.

3.2 Overall Architecture of MidAS System

The primary objective of the present research work is to design a robust and automated system to generate quality midsurface by leveraging feature information. The resultant midsurface is aimed at having minimum errors, such as gaps, overlaps, etc. and represents the input model shape. Such midsurfaces can be used for variety of downstream applications such as CAE analysis of thin-walled parts, shape matching, retrieval, etc.

Figure 3.1 shows the architecture, overall work-flow and various constituent modules of the **MidAS** system. Each module is implemented using Application Programming Interfaces (APIs) provided by Autodesk Inventor. CAD Models shown at each stage demonstrate the transformation happening to the models. Following list elaborates each of these modules and their workings.

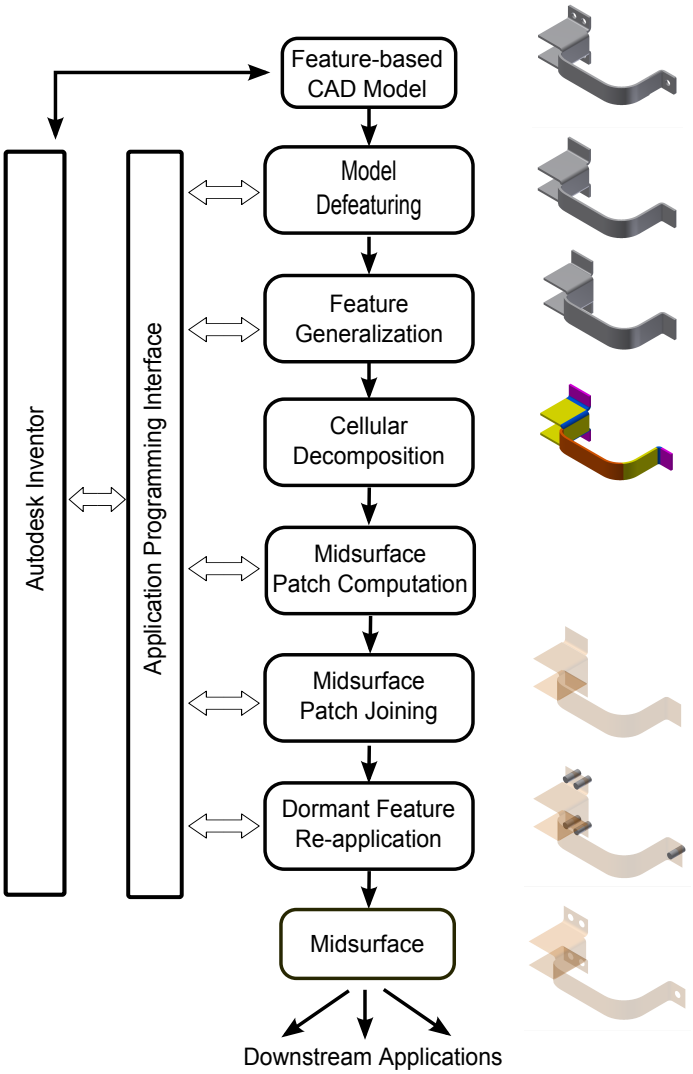


Figure 3.1: Overall System Architecture of **MidAS**

- **Input**: Input to **MidAS** is a CAD part model which is built by using a variety of sheet metal features. The model is represented as a feature history tree.
- **Model Defeaturing**: This module defeatures the input CAD model by removing irrelevant features. The criterion for deciding relevance is based on newly devised approaches based on sheet metal feature's taxonomy and remnant feature volumes. Apart from these two approaches, in a third approach, relevant but negative features, such as Holes, Cutouts, etc. are also removed after storing their tool-feature-bodies. These features are called as Dormant features. Removal of dormant features further simplifies the model, thus making further midsurface computation stages easier. Once midsurface is computed, the stored dormant feature bodies are re-applied back so that they preserve their impression on the midsurface. Detailed approach of this module is presented in chapter 4.
- **Feature Generalization**: A typical sheet metal part is modeled through a variety of sheet metal features with feature parameters. Though these features meaningfully represent the context of sheet metal domain their sheer number and variety

can make the midsurface generation a complex task. Simplifying them further will obviously make midsurface computation easier and quicker. This module, with a set of algorithms transforms the actual feature tree of the model into the one represented by very basic and fundamental modeling features such as Extrude, Revolve, Sweep and Loft, along with boolean operations. Since such a model essentially contains only Loft-equivalent features combined with boolean operations, the complexity of the midsurface computation drastically reduces. Detailed approach and methodology of this module is presented in chapter 5.

- **Cellular Decomposition**: This module takes input a simplified and generalized feature based CAD model and decomposes it into finite number of manageable volumes called "cells". These cells typically represent primitive shapes of solid, each with a Loft-equivalent owner feature. The connectivity of these cells is captured in a graph based data structure. Graph nodes are classified into midsurface patch generating nodes, called as patch nodes and midsurface patch joining nodes called as junction nodes. Cells pointed by patch nodes are called as solid cells. Cells pointed by junction nodes are called as interface cells.
- **Midsurface Computation**: Midsurface patches are generated for each solid cell, either by offsetting the profile face or by first generating midcurve from the profile followed by lofting it. Incident midsurface patches are subsequently joined within interface cells. Details of the algorithms are presented in Chapter 6.
- **Dormant Feature Re-application**: Dormant feature tool bodies are reapplied to bring back the relevant negative features onto the midsurface.
- **Validation**: The output midsurface is validated by newly devised approach topological validation, in which predicted topological entities of midsurface are matched with the topological entities of the actual output midsurface. Any mismatch tells the presence of errors such as missing faces, gaps, etc. This approach is only a theoretical proposal and has not been implemented in **MidAS**.

3.3 System Specifications

MidAS has been designed and developed in a modular fashion employing Object-Oriented (OO) Programming methodology. It is implemented using OO programming languages, such as VB.Net and C#.Net. **MidAS** interacts with Autodesk Inventor modeler through its published APIs to query and reason the model data as well as to render the output.

Prototype implementation has been done on Intel 64 bit processor PC. Many of the example parts used to demonstrate the concepts, have been borrowed from GrabCAD® site (http://www.grabcad.com).

3.4 Implementation Approach

Autodesk Inventor APIs were chosen for **MidAS**'s implementation, because of availability of the student version for free and mature APIs over releases. Autodesk also provides free technical support. The student version also comes with own in-built CAE module, where the output midsurface can be tested.

CAD APIs are typically used to write custom programs in various ways such as Add-Ins, External programs, etc. **MidAS** has built various External programs using APIs. Implementation of **MidAS** consists of neatly encapsulated software modules by using Object Oriented classes, that interact with each other through well defined interfaces. Core and internal algorithms are hidden from other classes to maintain security as well as to facilitate internal changes without disturbing the users of the classes. The modules are as follows:

- **Model Defeaturing**: VB.Net OO class Defeaturer encapsulates the logic of iterating over input sheet metal CAD features and removing the irrelevant ones. It exposes only the driver *Run*() function. This function does the actual defeaturing and shows performance statistics. The defeatured model is saved with "_Defeatured" extension. Tool bodies of dormant features are computed and stored to be used later for reapplication on the midsurface.

- **Feature Generalization**: VB.Net OO class Generalizer encapsulates the logic of iterating over "_Defeatured' sheet metal CAD features and transforming them into generalized loft equivalents. It exposes only the driver *Run*() function. The output generalized $\mathcal{ABLE}$ model is saved with "_Able" extension.

- **Cellular Decomposition**: CAD model with "_Able" extension is decomposed into cells, each having Loft-equivalent owner feature(s). Classes such as Graph, Node and Edge are used to populate the graph with each Node pointing to the cells. Edges are formed based on the connectivity between nodes. Nodes have attribute showing whether they are patch generating nodes or junction nodes. The decomposed model is saved with "_Decomposed" extension.

- **Midsurface Computation**: VB.Net program takes "_Decomposed" extension CAD model and iterates over patch creating nodes of the graph. It computes midsurface by first computing the midcurve of the owner feature's sketch and then lofting it.

- **Midcurve Computation**: Midcurve computation program is called from patch-creation module as mentioned above, by passing sketch as input, resulting into midcurves as output.

- **Midsurface joining**: After midsurface patches are ready, the junction nodes join the incident patches. Finally *Stitch* function is called just to join together already well-placed midsurfaces.

- **Dormant Re-application**: Stored dormant tool bodies are re-applied on the midsurface to pierce them through. The output midsurface is saved with "_Midsurface" extension.

The output midsurface is then sent for shell meshing for CAE analysis.

3.5 Scope of System

Although Autodesk Inventor exposes most of the functionality via its APIs, it does not do so fully, due to proprietary reasons. So, all the modeling interactions that are possible in the interactive mode are not always available via APIs. This puts some restrictions on the functionality that can be developed using APIs. It is however noted that those limitations do not anyway dilute the principles and objectives of the research. That is if the CAD modeler later provides the required capabilities through APIs, the research coverage can be extended.

MidAS has been implemented to cover sheet metal feature based CAD model as input. The model is considered to be built using the library of sheet metal features provided by Autodesk Inventor's sheet metal part modeling environment.

3.6 Overall Working of System

Overall working of the **MidAS** system constitutes the following steps in sequence:

- The input sheet metal CAD model is either prepared afresh in Autodesk Inventor or already created model is opened in the sheet metal modeling environment.

- Defeaturing functionally is invoked, which when run, removes all the irrelevant features and stores dormant tool bodies.

- Generalization functionality is invoked on the saved defeatured model. Each sheet metal feature gets converted to the corresponding Loft equivalent features, thus forming a generalized CAD model. Shape and size of the model remains as is, only the feature tree gets transformed.

- Decomposition functionality decomposes the generalized CAD model into list of solid cells. Cells are connected to form a cellular graph. Nodes of the graph are classified as patch nodes and junction nodes. Patch nodes point to solid cells and junction nodes point to interface cells.

- Midsurface generation module starts computing midsurface patches from solid cells and joins them in the interface cells.
- Dormant feature tool bodies are re-applied onto the midsurface.
- The midsurface is validated for topological and geometrical correctness.

This chapter has presented an overview of the newly proposed system called **MidAS** and its architecture. It has provided a brief review of the functional modules, the scope and the system specifications. Finally overall working of **MidAS** has been outlined in the step-wise manner.

Following chapters present, in detail, the design and development of modules to process the CAD model by various algorithms to finally compute well-connected midsurface with a validation strategy.

Chapter 4

Defeaturing of Sheet Metal Feature-based CAD Model

4.1 Introduction

The previous chapter presented an overview of the newly proposed system called "**MidAS**" and its architecture. This chapter provides details of its first module, called Model Defeaturing, i.e. the algorithms proposed for defeaturing sheet metal feature based CAD model.

Defeaturing is one of the most popular approaches of simplifying CAD models for CAE analysis. CAD models contain various details (or features in case the CAD model is feature based) which are introduced due to requirements of various downstream applications such as CAM, Visualization, etc. Such detailed models are often not needed for CAE, especially for quicker validations at the early stages of design. Defeaturing is the process of removing irrelevant details or features in the context of application to generate the simplified model, called "gross shape". It is the principal shape that "represents" the input shape, but with lesser features. In CAE analysis, the finite element mesh generated on the gross shape that is obtained by defeaturing has far lesser number of nodes compared to the original input CAD model. Lesser number of nodes means lesser number of degrees of freedoms (DoFs) for solving the CAE equations, thus lesser computations and quicker analysis results.

Figure 4.1a shows a CAD model of a Rotor Brake part. It has about 50 distinct features and all of them are not relevant for the thermal analysis. After defeaturing, i.e. after removing small holes, depressions, cutouts, etc. the output gross shape is shown in Figure 4.1b. Gopalakrishnan [18] reported that, in this particular example, DoFs got reduced from 150,000 to 25,000 after defeaturing, thereby speeding up the computation significantly.

Apart from CAE, gross shapes are also used in shape matching & retrieval, faster visualizations, hiding proprietary details, quicker transmissions across the networks,

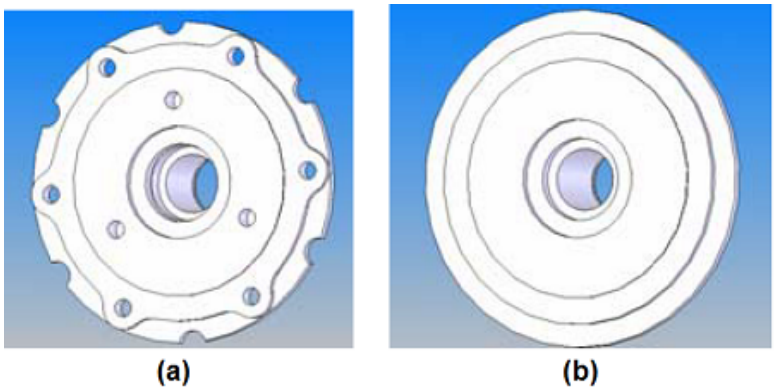


Figure 4.1: Gross Shape of Rotor Brake (Source: Gopalkrishnan [18])

etc. Reason being, it is far more efficient to work on the gross shape (which retains the important shape characteristics) than the detailed input CAD model. Thus, due to these advantages, defeaturing is found useful in a wide variety of applications.

The core decision in any defeaturing approach is the criterion of 'irrelevance' of a detail or feature in the context of application, which is CAE in this case. For example, for structural CAE analysis, following could be the criteria:

- Remove small fillets & chamfers. Results of the analysis will show increase in the stresses in the gross shape compared to the original input model. So the results will be conservative and safer for further part design steps.
- Do not remove features in the load path and near Boundary Conditions. They are critical to the results and are not to be removed even if they are eligible for removal due to smaller sizes.

The present research work takes sheet metal feature-based CAD model as input for computing midsurface. Hypothesis 2 which says that the gross shape enhances definitiveness of getting a quality midsurface, is demonstrated in this chapter. Following sections present systematic study of sheet metal feature characteristics to decide the eligibility of some of them for removal.

4.2 Need for Defeaturing

Literature survey Section 2.3 has reviewed various reported approaches of defeaturing. Subsection 2.3.4 concludes that these approaches did not adequately leverage the feature information and to a large extent did not consider application domain.

The present research focuses on defeaturing by leveraging feature information, of sheet metal features based CAD models. Lesser the features the model is built with, retaining the gross shape, easier the computation of midsurface. Following experiment assesses this intuition, which has been formalized in the Hypothesis 2 (Chapter 2).

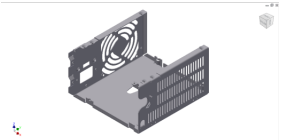
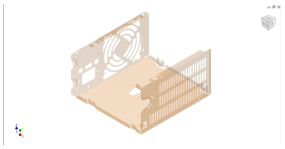
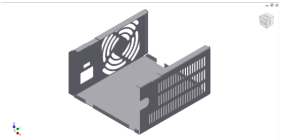
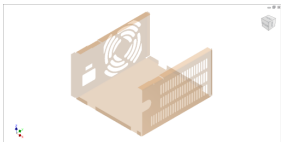
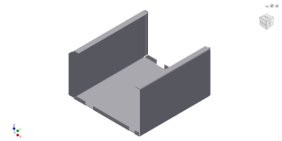
Part and actions	**CAD Model**	**Generated Midsurface**	**Errors in Midsurface**
Input model			Gaps and missing surfaces
Defeaturing of the small irrelevant features.			No major errors. Good midsurface.
All negative features suppressed.			No errors but midsurface lacks important shapes.

Table 4.1: Effect of Defeaturing on Midsurface Generation

Table 4.1 shows results of the experiment of studying the effect of defeaturing on the quality of the midsurface. The first row shows input CAD model and its midsurface computed using a commercial CAD-CAE system. The output midsurface shows various errors such as gaps and missing surfaces, as shown amplified in the picture under 'Quality of Midsurface' column. The second row shows the effect of defeaturing, i.e. the removal irrelevant features, such as smaller holes, corner rounds, rips, hems, etc. It does not remove bigger holes, cutouts, flanges, grills, etc. The resultant midsurface does not show any major errors and is of good quality. The third row shows the effect of substantial defeaturing, i.e. removal of all the negative features, even if they are relevant. So all cutouts, patterns, grills, etc. are removed. The output midsurface, although has no gaps, or missing surfaces, lacks presence of the relevant negative features. Such defeaturing needs to be avoided as the midsurface does not represent the correct gross shape. Although this experiment validates the Hypothesis 2 to be true, but suggests judicious selection of features for removal.

The present research work, however, proposes use of the substantial defeaturing approach, i.e. the third row shown in Table 4.1. Removal of all the negative features simplifies the computation of midsurface to a large extent. The proposed approach overcomes the problem of missing relevant negative features on midsurface, by storing the tool bodies of those negative relevant features and bringing them back after computing the midsurface for piercing into it. With this arrangement, computing midsurface becomes easier and the output midsurface is not devoid of the impressions of relevant negative features.

Following section proposes a multi-phase approach for computing the 'gross shape', which in turn, helps compute a quality midsurface.

4.3 Proposed Approach to Compute Gross Shape

In the context of the present research work, formulating rules for identification of the irrelevant features is the most critical step that affects the output gross shape. Researchers have measured the relevance of each feature by evaluation metrics [97].

The evaluation metrics proposed in the present research is divided into three criteria, viz. application context-specific criteria, geometric reasoning-based criteria and dormant features criteria.

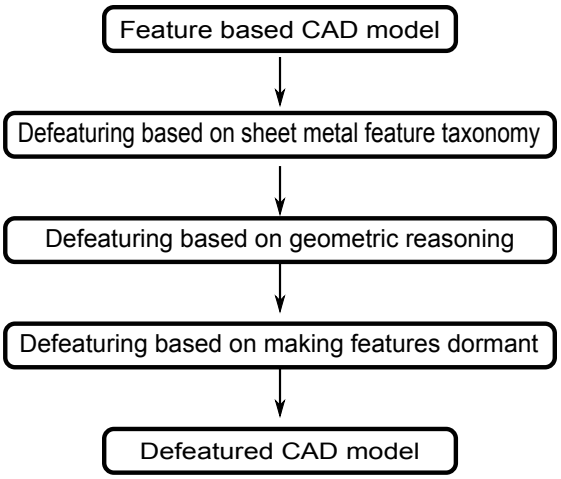


Figure 4.2: Overall Defeaturing Approach

Figure 4.2 shows phase-wise processing of the input sheet metal feature based CAD model by these criteria. Essentially these criteria apply various rules pertaining to application context, geometric reasoning and negative features, respectively. These rules are applied to identify the criticality of the features in the CAD model in the context of midsurface generation and attempts are made to remove not-so-relevant features from it. A threshold value is computed and provided to the overall approach. It denotes the "size" below which a feature is considered as irrelevant. The process thus simplifies the CAD model by defeaturing it while maintaining the gross shape intact. The phases of the proposed defeaturing process are explained below.

- **Phase I - Sheet Metal Taxonomy based**: The sheet metal CAD model feature tree is traversed and the candidate features for suppression are identified based on criteria based on the newly proposed sheet metal features taxonomy. For other domains or applications, this phase can be customized by employing application context specific taxonomies.
- **Phase II - Remnant Features**: This phase is generic and starts with the final Brep for identifying the remnant portions of the features. Those whose sizes are below the threshold are identified for suppression. One can customize the threshold based on engineering judgment appropriately.

- **Phase III - Dormant Bodies**: In this phase all "negative" features, even though they are relevant and thus have not been identified by the above two phases, are selected for removal, temporarily. Before removal, their feature bodies are cached/stored and are later used for re-applying on the output midsurface.

In the system implementing the proposed approach, it is important to note that, during defeaturing, even though the irrelevant features are said to be removed, they are discarded only temporarily and are not deleted permanently so that, in case of failure, they can be brought back and the original input CAD model to that stage can be restored.

Following sections present the algorithms for these phases in details.

4.4 Defeaturing Based on Sheet Metal Features Taxonomy

CAD model under consideration is built by a number of sheet metal features such as Wall, Flange, Bend, etc. and are represented as a feature tree. This feature tree is traversed and the candidate features for removal are identified based on a criteria using the newly proposed sheet metal features taxonomy.

Figure 4.3 shows the proposed taxonomy which classifies sheet metal features and suggests their relevance with respect to maintaining the gross shape. In comparison with the other sheet metal features classifications, such as for features recognition [98, 99] and process planning [100], the proposed taxonomy is different in a way that the classification is done based on the criticality of a particular feature to the gross shape.

4.4.1 Sheet Metal Features Taxonomy

A thorough analysis of the inputs from various surveys with engineers and experts in the field to gauge relevance of sheet metal features with respect to the gross shape. Proposed taxonomy (Figure 4.3) is the result of this analysis.

Taxonomy is represented by "vocabulary" and "structure" [101]. It is a scheme to represent and classify features for a particular purpose [102]. Figure 4.3 shows the classification of sheet metal features for the purpose of computing gross shape.

- **Primary Features**: These features mainly contribute to the gross shape. Examples are Face-Wall, Flange, Bend, etc.
- **Secondary Features**: Relevance of these features is based on the relative size with respect to the size of the whole model. Examples are Stamping, Cutout, Emboss, etc.
- **Tertiary Features**: These features are irrelevant to the gross shape irrespective of their size relative to the size of the whole model. Examples are Lip, Rest, etc.

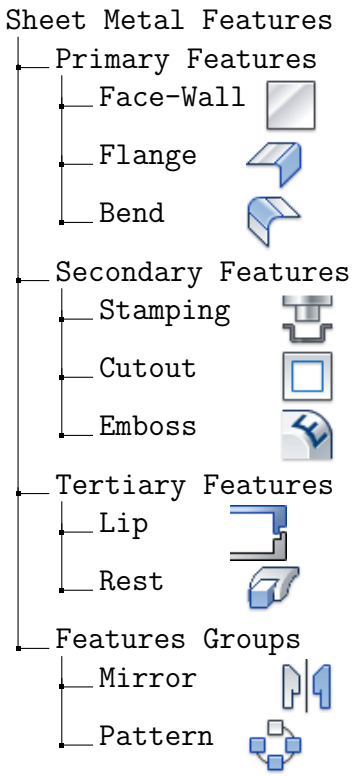


Figure 4.3: Sheet Metal Features Taxonomy (Icons source: Autodesk Inventor [19])

- **Feature Groups**: These are feature collections. Their relevance is assessed as a collection, similar to the secondary features. Examples are Mirror, Patterns, etc.

Figure 4.3 shows the proposed taxonomy classifying sheet metal features into categories such as Primary features, Secondary features, Tertiary features and Feature groups, based on their sheet metal domain specific characteristics. For defeaturing, each feature category undergoes its own rule for eligibility for removal during defeaturing process. Those rules are elaborated in detail in Section 4.4.2.

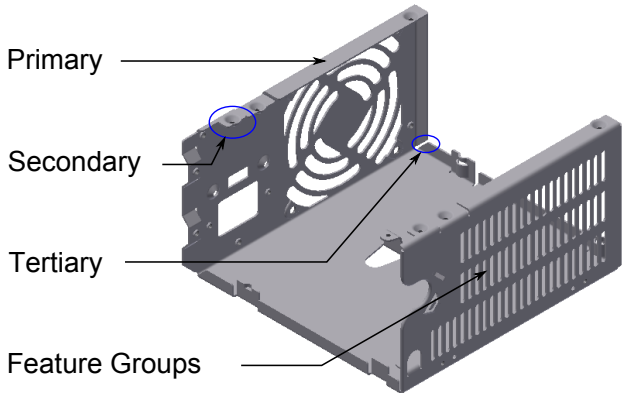


Figure 4.4: Sheet Metal Features Based on Proposed Taxonomy

Figure 4.4 shows example part with various sheet metal features classified as per the proposed taxonomy.

4.4.2 Defeaturing Algorithm Based on Sheet Metal Features Taxonomy

This subsection details the steps used to identify the removable sheet metal features based on the taxonomy proposed in Section 4.4.1. Threshold value is computed based on the experimentations done [103] to assess the effect of different threshold values on the quality of resultant midsurface output. Parameter D, which is used as a size criteria for deciding irrelevance of a feature, is defined as threshold %age (p) of the size of the CAD model. Size of the CAD model can be defined in multiple ways, such as, size of Bounding Box, volume, etc. Bounding box as a measure of size, is quick to compute but not accurate. Volume is more accurate than Bounding box but is compute intensive, especially for Brep based CAD model, where the model is composed of faces. Sum of areas of the faces is both, reasonably accurate and easier to compute the size. Thus the present research measures size of the model as sum of the areas of all the faces. The %age value for threshold is typically based on expertize, application context and engineering judgment. Various experiments were conduced to correlate effect of %age value of threshold on the gross shapes. Table 4.1 summarizes the results and more details are presented in Appendix B. Based on those experiments typical threshold values used in the present research work are 5-10%.

Defeaturing rules based on sheet metal features taxonomy are enumerated below:

1. **Primary Features**: Not removed as these features mainly contribute to the gross shape. Examples are Face-Wall, Flange, Bend, etc.
2. **Secondary Features**: Are removed based on the relative size with respect to the size of the whole model. Features smaller than the threshold are removed, whereas bigger ones are retained.
3. **Tertiary Features**: Are removed irrespective of their size relative to the size of the whole model.
4. **Feature Groups**: Are removed based on the collective relative size with respect to the size of the whole model.

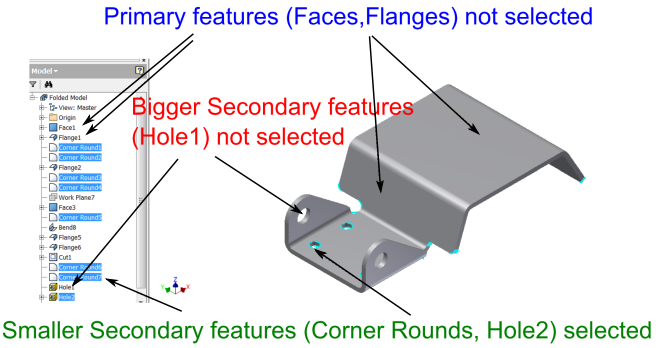


Figure 4.5: Defeaturing Based on Proposed Taxonomy

Figure 4.5 shows an example sheet metal part CAD model, with taxonomy based

features categories shown, along with their defeaturing rules. Primary features such as Walls (Faces), Flanges are not removed. Secondary features whose face-size is less than the threshold are removed, but the larger ones are retained. With threshold as 5 %, holes smaller than 5 % of the model faces area, such as Hole2 and CornerRounds are selected but not the bigger holes such as Hole1.

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** $f_i = currentFeature() != null$ **do**
5: **if** $f_i \rightarrow isPrimaryFeature()$ **then**
6: continue
7: **else if** $f_i \rightarrow isTertiaryFeature()$ **then**
8: $sl \rightarrow add(f_i)$
9: **else if** $f_i \rightarrow isGroupFeature()$ **then**
10: **if** $f_i \rightarrow combinedArea() < D$ **then**
11: $sl \rightarrow add(f_i)$
12: **end if**
13: **else**
14: **if** $f_i \rightarrow area() < D$ **then**
15: $sl \rightarrow add(f_i)$
16: **end if**
17: **end if**
18: **end while**
19: $sl \rightarrow removeAll()$
20: $rebuildModel()$

Following are the steps taken for defeaturing CAD model based on sheet metal features taxonomy. Algorithm 1 presents the same in pseudo-code form.

1. The threshold %age value is chosen. The threshold size is the threshold percentage times of the size of the model. The size of the model is computed as the total of areas of all faces of the model. It is denoted as "face-size" of the model. Similarly, size of the feature is denoted as "face-size of the feature" and is the sum of area of the faces of the particular feature (Algorithm 1: lines 1-3).
2. The model feature tree is traversed and the current feature is applied with the defeaturing rules based on the newly proposed taxonomy (Figure 4.3, Figure 4.5) (Algorithm 1: loop 4 to 18).
3. The Primary features are not selected for removal so get skipped and do not get

added to the removable-features list (Algorithm 1: lines 5-6).

4. If the current feature is a “Tertiary” feature, then it is added directly to the removable-features list (Algorithm 1: lines 7-8).
5. If the current feature is a “Group” feature, then it’s total face-size of the constituent features is checked against the threshold and if it is less then it is added to the removable-features list (Algorithm 1: lines 9-12).
6. If the current feature is a “Secondary” feature, then it’s face-size checked against the threshold and if it is less then the feature is added to the removable-features list (Algorithm 1: lines 14-16).
7. Once all the features in the model are traversed, the features in the removable-features list are removed and the model is regenerated (Algorithm 1: lines 19-20).

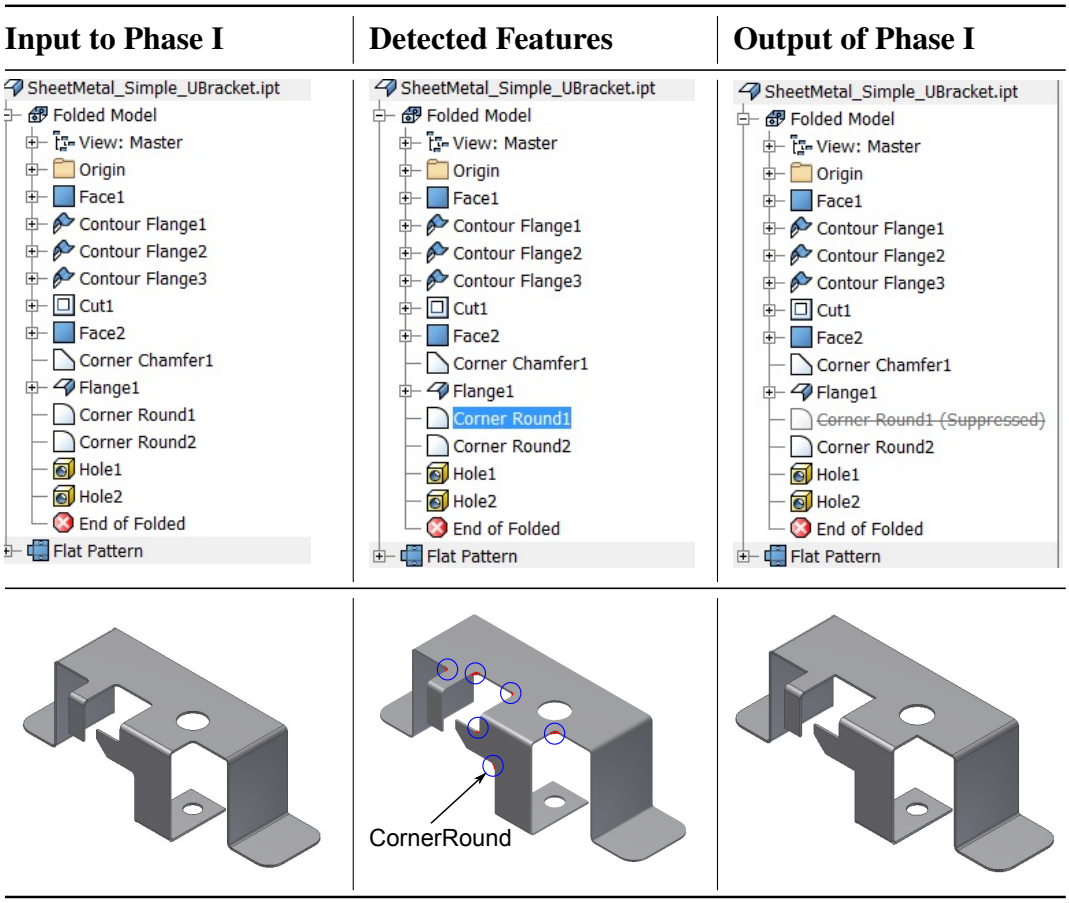


Figure 4.6: Phase I: Defeaturing Based on Feature Taxonomy and Size Threshold Approach

Figure 4.6 shows the process pictorially. The first column, called “Input to Phase I” shows input sheet metal features CAD model of a bracket part, along with its feature tree. The second column, called “Detected Features” shows the candidate features selected, such as ‘Corner Round1’, on the model as well as in the feature tree, 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.

With the Phase I over, sheet metal domain specific defeaturing is complete. This

phase can be further enhanced by either customizing the threshold and/or by incorporating more features in the taxonomy based on further surveys and experimentations. It can be customized further, for different application domains, such as Injection Molding, by incorporating taxonomy specific to those domain.

Following section details out the Phase II, i.e. defeaturing process based on geometric reasoning.

4.5 Defeaturing Based on Geometric Reasoning of CAD Model Features

The sheet metal feature based CAD models used in the present research work are built using 'design-by-features' approach. At each feature step, the feature parameters first generate a feature primitive solid, known as "tool-body". Boolean operation is performed between operands: the existing model at that stage and the tool body. During this operation, some portion of the tool-body-portion gets consumed within the existing model and the remaining portion of the tool body appears as a newly added feature in the CAD model.

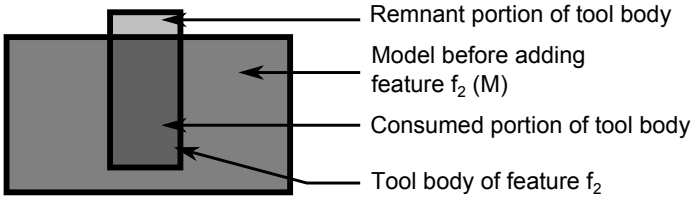


Figure 4.7: Remnant and Consumed Portions

Figure 4.7 shows boolean operation between the existing CAD model denoted by M and the tool body of feature f_2. Some portion of the tool body gets consumed and is merged with the model whereas some portion remains outside. The portion remaining outside is referred as "remnant feature" portion. This phase involves the geometric reasoning to identify remnant feature portions on the input Brep CAD model.

Past attempts such as by Russ [53], used full feature tool-body to determine the candidature for removal during defeaturing. This obviously gives incorrect results as the CAD model actually retains only the remnant portion of the feature. Size of only remnant portion should be considered for deciding the candidature for removal. Thus the present research uses the size of the remnant portion for deciding removal of a feature and the algorithm to do this is presented below.

4.5.1 Defeaturing Algorithm Based on Remnant Feature Portions

The input to this phase is the output CAD model from Phase I. In contrast to phase I, this phase does not involve traversal of the feature tree. Instead, it traverses faces of

the final model i.e. the Brep model available at the end of the feature tree. As these faces have remained in the model till the end, they are termed as 'remnant' faces. The Brep CAD solid model is termed as 'final body'. Whenever a feature is added, it adds new faces to the existing Brep model. These new faces store information of the feature which created it, which is termed as the "owner feature". The owner feature information is stored on all faces of the Brep as attributes, typically referred as "origination info".

The intent of this phase is to group all the remnant faces belonging to a particular feature. This group is termed as 'FaceGroup'. Then identify their respective feature owners, and decide their removability on the size of their remnant portions. The size of a FaceGroup can be calculated by various approaches like Influence Volume [59] or the union of bounding-boxes, etc. The present work uses summation of the area of the remnant faces as the 'Size' criterion and is called "face-size". The choice of area as the size criteria is based on ease of computability and reasonable accuracy to represent the remnant portion.

Following steps present details of the algorithm used to identify remnant faces and thereby remnant features, to be removed based on the remnant portion of the tool body (Figure 4.7) (Algorithm 2).

1. Remnant faces of the final body are traversed. (Algorithm 2: loop 1 to 15).
2. For each remnant face, its owner feature is retrieved (Algorithm 2: line 2).
3. Existing FaceGroups are searched for matching owner feature.
4. If a match exists, the face is added to that FaceGroup (Algorithm 2: lines 4-8).
5. If a match is not found, then a new FaceGroup is created, the remnant face is added to it, it owner feature is set as the owner of the FaceGroup and this new FaceGroup gets appended to the list of FaceGroups (Algorithm 2: lines 9-13).

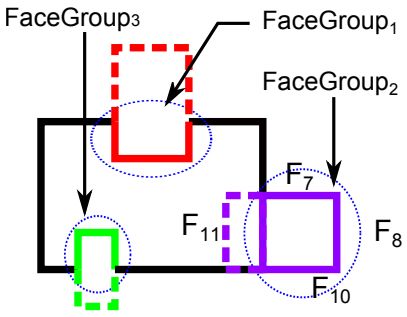


Figure 4.8: FaceGroups

Table 4.2: FaceGroups Details

FaceGroup Id	**Face-size** (mm^2)	**Owner Feature**
$FaceGroup_1$	0.25	$Extrude_2$
$FaceGroup_2$	0.25	$Extrude_3$
$FaceGroup_3$	0.125	$Hole_1$

Figure 4.8 shows a schematic of existing Brep CAD model with 3 features booleaned to it. Thick lines are used to represent the remnant faces whereas dotted lines are used to represent the consumed faces. FaceGroup1 and FaceGroup3 are depression features, i.e. negative features, whereas FaceGroup2 is a protrusion feature i.e. a positive feature. In FaceGroup2, F_{11} is a consumed face whereas Faces F_{10}, F_8, F_7 are the remnant faces. So, FaceGroup2 represents remnant portion of

the owner feature Extrude_3.

6. Face-sizes of the FaceGroups are computed. Table 4.2 shows face-size (in mm^2) along with owner feature of each FaceGroup.
7. A FaceGroup is removed if its face-size is below the threshold value (Algorithm 2: lines 16-18). The threshold used is same as the one used in Algorithm 1: lines1-3.
8. Once all the FaceGroups are assessed, removable-features are removed from the feature tree and the model is rebuilt (Algorithm 1: lines 20-21).

Algorithm 2 Remnant Feature Method

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

1: **while** $F_i = currentFace() != null$ **do**
2: $feat = F_i \rightarrow owingFeature()$
3: $addedFlag = false$
4: **while** $cl_j = currentFaceGroup() != null$ **do**
5: **if** $cl_j \rightarrow owingFeature() == feat$ **then**
6: $cl_j \rightarrow add(F_i), addedFlag = true$
7: **end if**
8: **end while**
9: **if** $addedFlag == false$ **then**
10: $cl_n = newFaceGroup()$
11: $cl_n \rightarrow owingFeature() = feat$
12: $cl_n \rightarrow add(f_i)$
13: **end if**
14: **end while**
15: **while** $cl_k = currentFaceGroup() != null$ **do**
16: **if** $cl_k \rightarrow calculateSize() < D$ **then**
17: $sl \rightarrow add(cl_k)$
18: **end if**
19: **end while**
20: $sl \rightarrow removeAll()$
21: $rebuildModel()$

Figure 4.9 shows the process pictorially. The first column, called "Input to Phase II" shows the sheet metal features CAD model of a bracket part partially defeatured in the Phase I, along with its feature tree. The second column, called "Detected Features" shows the candidate features selected, such as Holes, on the model as well as in the feature tree, based on Algorithm 2. 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. Third column shows the output of Phase II, showing the model with candidate features removed.

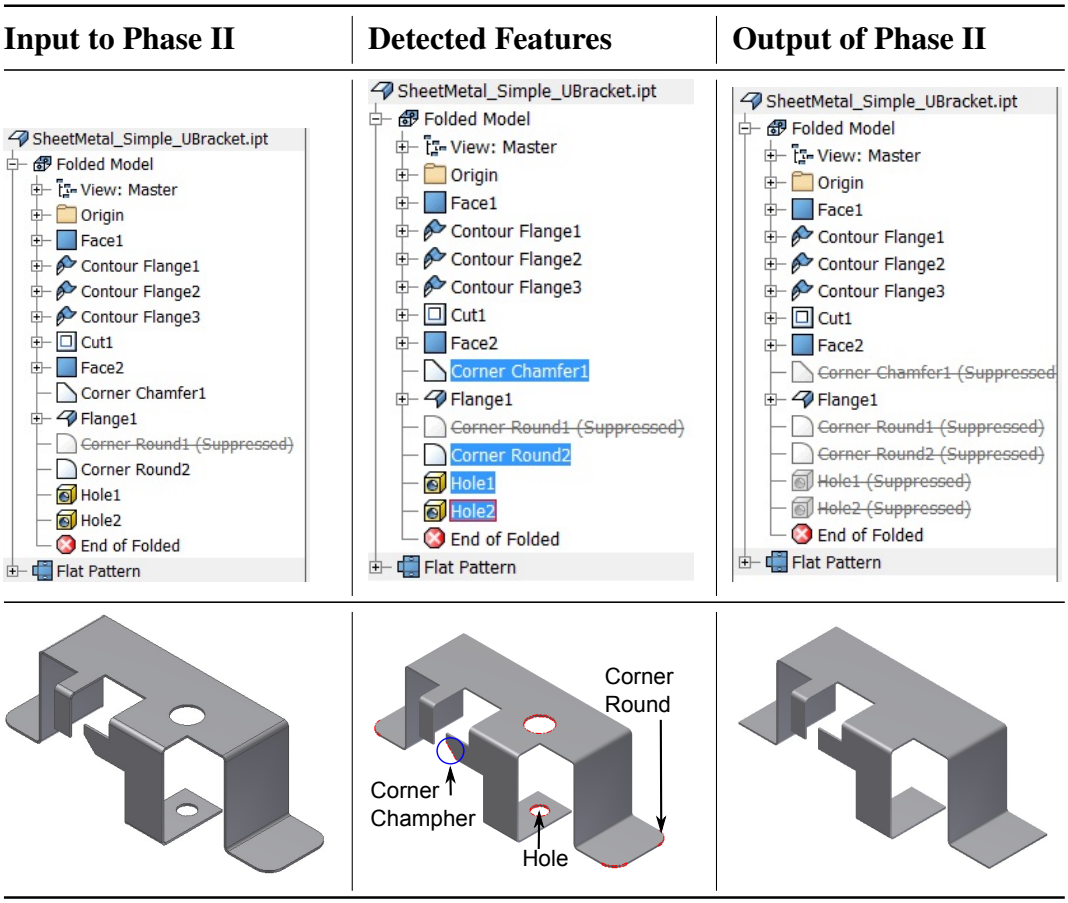


Figure 4.9: Phase II: Defeaturing Based on Remnant Feature Approach

After the first two phases, the given example of bracket model shows 50% reduction in the number of features and 17% reduction in the number of faces, at 5% threshold value. The resulting gross shape has retained all important features necessary for the computation of a quality midsurface.

4.6 Defeaturing Based on Dormant Feature Tool-bodies

Bulk negative features, such as large Holes, Cutouts can be potentially problematic in further processes used in the midsurface computation, such as feature generalization (Chapter 5) and cellular decomposition (Chapter 6). But they can not be removed because they are necessary constituent of the gross shape.

The results shown in Table 4.1 demonstrate that removal of negative features, even though they are relevant, can help computation of midsurface. But the drawback is, these features, being relevant would be missing on the midsurface. The present research work proposes to gain the computational advantage but at the same time does not want

to loose the relevant features. This proposed approach is called as the Dormant Features approach. In this approach, relevant negative features are removed temporarily, but while doing so their feature bodies (called "tool-bodies") are preserved for later usage. After computing midsurface from such highly defeatured model, the preserved tool bodies are brought back and are re-applied on the midsurface. Thus, effect of those important, relevant negative features on midsurface is not lost. However temporary removal does substantially simplify the model which make midsurface computation quite convenient. Following section elaborates Phase III, the dormant features approach.

4.6.1 Defeaturing Algorithm Based on Dormant Features

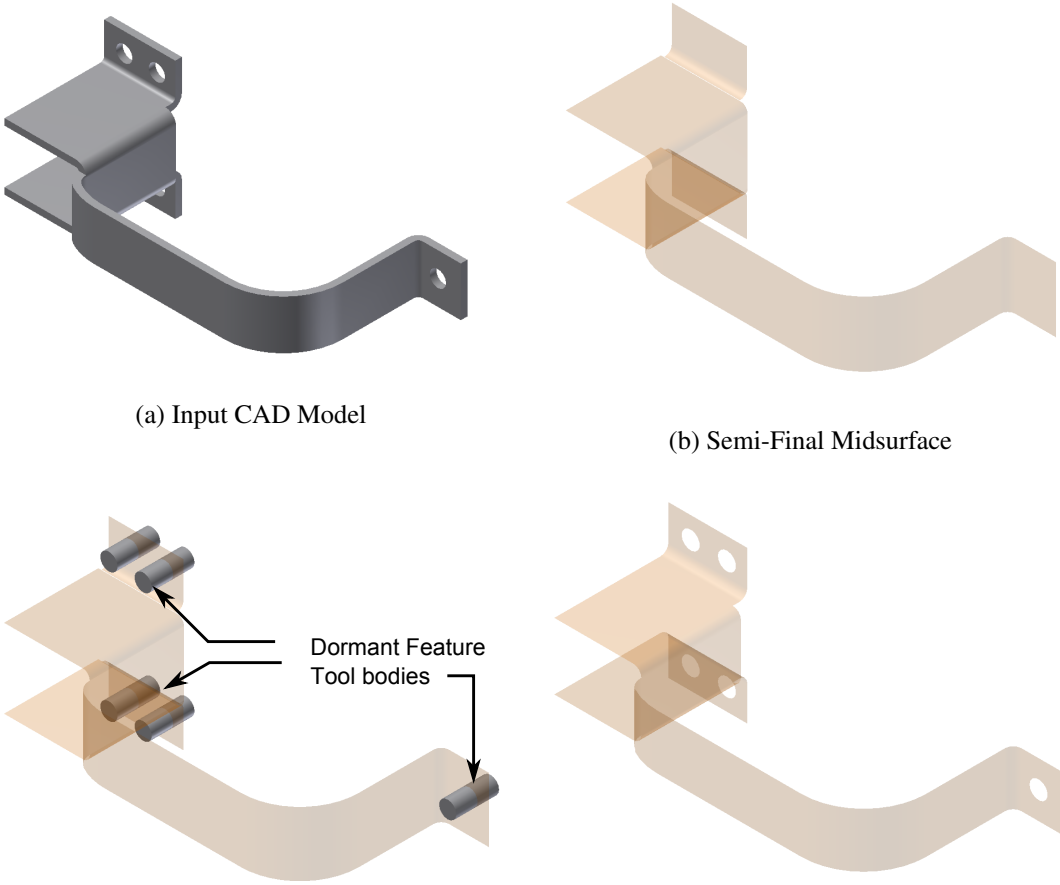
At this stage, Phases I and II are already over. All the small, irrelevant features are already removed. All the remaining features now, are the relevant features. These can include the negative features as well. In the steps mentioned below, in-spite of being relevant, the large negative features are temporarily removed, after storing their tool bodies. These bodies are kept dormant to be used later for piercing on the midsurface. Algorithm 3 presents pseudo code of the approach and has been listed below:

Algorithm 3 Phase III: Dormant Features Identification

Require: A Sheet Metal FCAD model with access to the feature tree

1: **while** $f_i = currentFeature() != null$ **do**
2: **if** $f_i \rightarrow isNegativeFeature()$ **then**
3: $b_i = computeToolBody(f_i)$
4: $bl \rightarrow add(b_i)$
5: $sl \rightarrow add(f_i)$
6: **end if**
7: **end while**
8: $sl \rightarrow removeAll()$
9: $rebuildModel()$

1. The input CAD model feature tree is traversed (Algorithm 3: loop 1 to 7).
2. The current feature is tested if it is a negative feature, like Hole, Cutout, Extrude with subtractive boolean, etc. (Algorithm 3 line: 2).
3. The tool body is computed using its feature parameters (Algorithm 3 line: 3).
4. The tool-body is stored in a list (Algorithm 3 line: 4).
5. The current feature is added to list for removal (Algorithm 3 line: 5).
6. Once all the features in the feature tree are traversed, the features from candidate features list are removed and the CAD model is rebuilt (Algorithm 3: lines 8-9).



(a) Input CAD Model

(b) Semi-Final Midsurface

(c) Dormant Tool Bodies Re-application

(d) Final Midsurface

Figure 4.10: Re-application of the Dormant Feature Tool-bodies on Midsurface

Figure 4.10 shows reapplication i.e piercing of the dormant feature tool bodies. Figure 4.10a depicts model before detection of the dormant features. Figure 4.10b shows midsurface computed after removal of the dormant features. The cached dormant feature tool bodies are brought back and are re-applied on the midsurface, as shown in Figure 4.10c. So, the presence of relevant negative features is made sure in the final midsurface as depicted in Figure 6.42b. Thus ensuring faithful representation of all the prominent features on the midsurface.

Table 4.3 shows a brief comparison of the present research work, as implemented in **MidAS**, with the other defeaturing approaches, such as, by Brian Russ [53], Sungchan Kim et. al [86] and Sang Hun Lee [45].

Table 4.3: Comparison of the Present Research with Other Methods

Methods	**Russ [53]**	**Kim [86]**	**Lee [45]**	**MidAS**
Input	Features	Brep	Features	Features
Approach	Suppresses if feature size is less than threshold.	Wrap-around. Negative volumes removed totally.	Feature volumes are reordered in sorted manner.	Suppress based on taxonomy and remnant volumes.
Advantages	Automatic identification of non-critical features	Does not need features; Works on Brep	Multiple defeaturing levels possible	More accurate criteria for defeaturing
Limitations	Limited to FEM as BC and Load path features are not suppressed.	Concave edge filling creates odd shapes. Principal shape is lost.	Due merged Cellular model, update capability is lost forever.	Taxonomy needs to be updated for new features

The comparison indicates that the defeaturing approach presented in this chapter significantly reduces the number of faces in the CAD model thus simplifying it considerably for the purpose of midsurface computation. It, however, still ensures that the gross shape of the input CAD model is retained so that underlying design intent is not lost.

In summary, Phase I removes features from the input CAD model based on the newly proposed sheet metal features taxonomy. The model gets further simplified in Phase II using remnant feature portions approach. Phase III simplifies the model further by completely removing all the negative features. After completion of all three phases of defeaturing the input sheet metal features CAD model is more or less reduced to its gross shape. This is optimum and effective level of defeaturing. Any further removal of the features will disturb the original gross shape intent.

Following section presents a newly proposed measure to assess the effectiveness of the defeaturing process.

4.7 Effectiveness of Defeaturing

Effectiveness of the defeaturing process is computed using a wide variety of methods, such as quantitative reduction, MAT, Mesh based, etc. They can be classified into input-based and output-based methods. In the input-based method, engineering judgment is used to set the initial defeaturing parameters, such as, size threshold, feature taxonomy, etc. and the output resulted is considered as the valid [104]. In the output-based method, some initial defeaturing parameters are set and the output is assessed against predefined benchmarks, such as, reduction in volume or number of faces or features, etc. The process is repeated till the benchmarks are achieved.

In the present research, the input-based method is used to assess the effectiveness of defeaturing. It is computed by measuring **Percentage reduction in the number of faces**. Measurements are done before and after a particular defeaturing phase. More the percentage, more effective is the defeaturing process. A measure of reduction in number of features can also be used in place of faces to form another criterion for assessing the effectiveness. The present research work collects following parameters to compute the effectiveness:

1. Total number of the faces in the original part (nF)
2. Number of faces remaining in the model after defeaturing (rF)
3. Defeaturing effectiveness (pR) (%)

$$pR = (1 - \frac{rF}{nF}) \times 100 \tag{4.1}$$

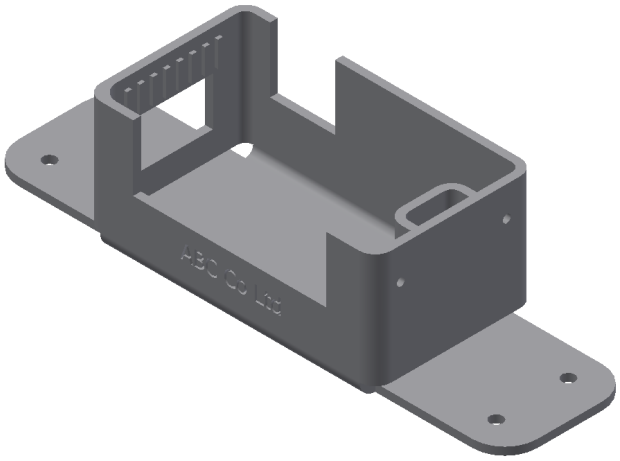
Defeaturing effectiveness (pR) gives quick idea of the order of magnitude of defeaturing.

Following section demonstrates efficacy of the proposed defeaturing approach by evaluating it against CAD model of a real-life test part.

4.8 Examples

This section exemplifies the defeaturing approach on a typical sheet metal part used as “Electronics Enclosure”. It is a typical sheet metal casing model used in electronics equipments.

Following are the steps through which defeaturing of “Enclosure” happens. The threshold (D) taken here is 5%.



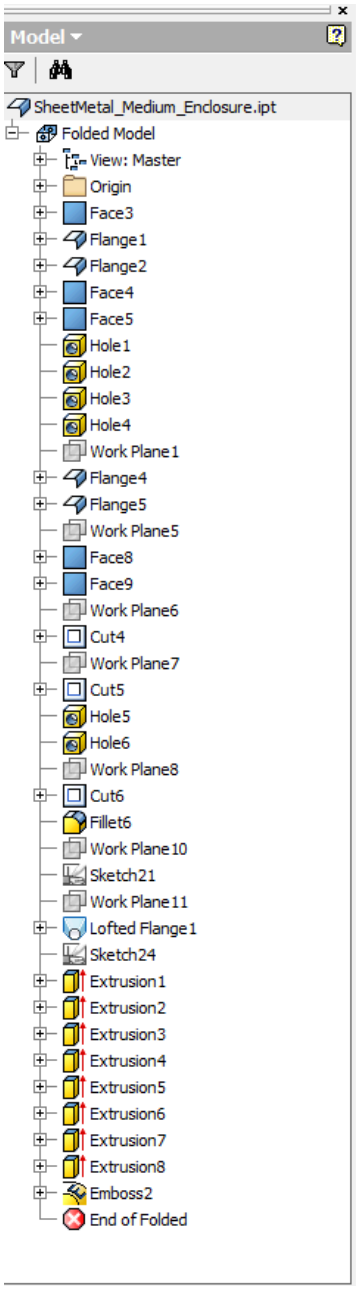
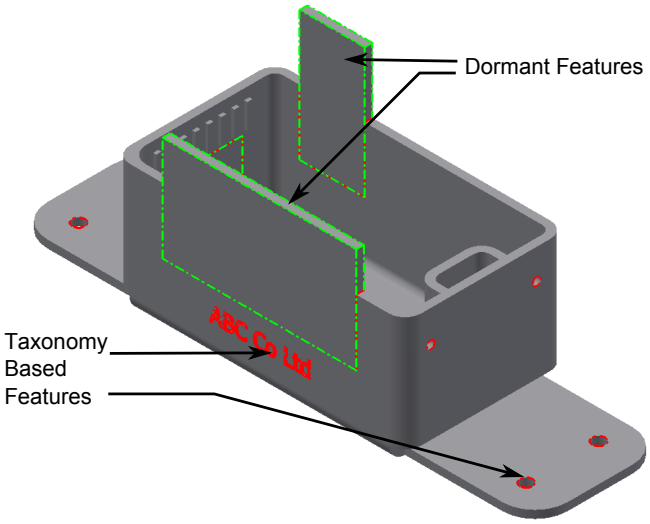


Figure 4.11: Input CAD Model

Figure 4.11 shows the input Enclosure part model its corresponding feature tree. It is modeled in Autodesk Inventor using sheet metal features such as Face(Wall), Flange, Bend, Emboss, array of Slots, etc.



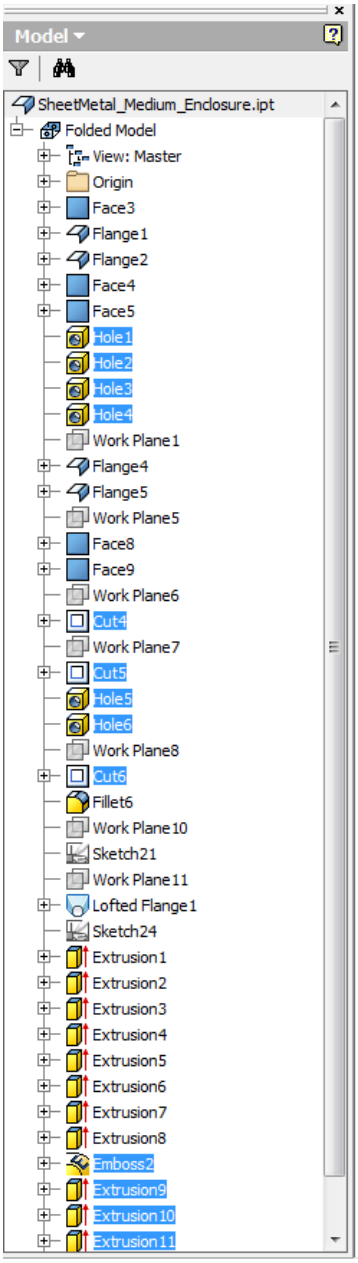
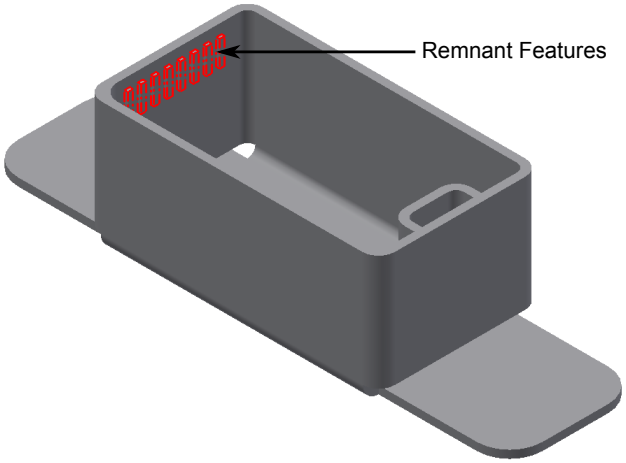


Figure 4.12: Features Identified for Removal Shown in Model and Tree

Figure 4.12 shows the selection of irrelevant sheet metal features based on newly proposed sheet metal feature taxonomy. It also shows the bulk negative features as dormant features.



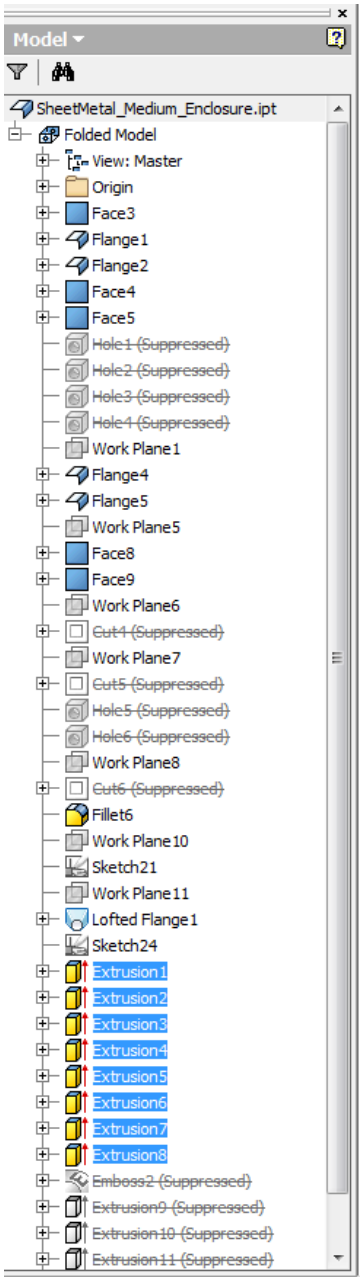


Figure 4.13: Removal of Features Based on Remnant Feature Approach

Figure 4.13 shows the selection of irrelevant sheet metal features based on remnant feature portions.

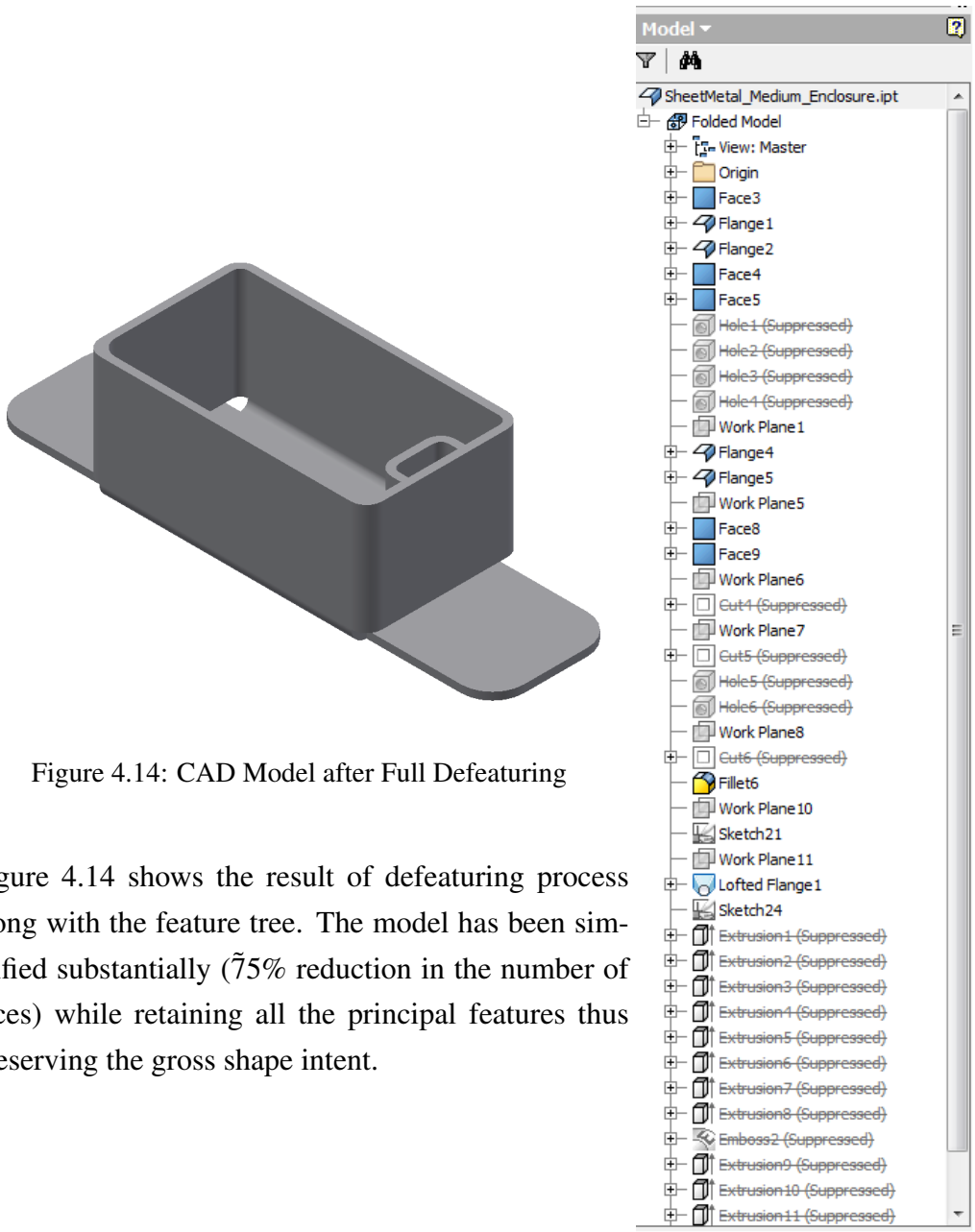


Figure 4.14: CAD Model after Full Defeaturing

Figure 4.14 shows the result of defeaturing process along with the feature tree. The model has been simplified substantially ($\tilde{}$75% reduction in the number of faces) while retaining all the principal features thus preserving the gross shape intent.

Effectiveness with 5% threshold, based on the criterion defined by Eqn. 4.1 is:

Table 4.5: Defeaturing Effectiveness Data

Number of Faces Before Defeaturing	**Number of Faces remaining after Taxonomy-Size Based Defeaturing**	**Number of Faces remaining at the End**
259	104	64

$$pR = (1 - \frac{64}{259}) \times 100 = 75.29\% \tag{4.2}$$

Table 4.5 shows the data collected at each phase. It can be seen that even after

huge reduction in the number of faces (75% seen in Eqn. 4.2), the overall shape of the enclosure is retained appropriately. This defeatured model is sent for further processing to compute a quality midsurface.

Following section presents conclusions of the process of investigation of problems and proposing solution for defeaturing a sheet metal feature based CAD model.

4.9 Conclusions

Literature reports that most of the defeaturing algorithms are based on the mesh or solid (Brep) as input. In these cases some kind of feature recognition needs to be performed first and then irrelevant features are identified and removed. With the availability of ready feature information in the feature based CAD models, it has become possible to leverage it for defeaturing purpose effectively.

The present research proposes to leverage feature information and presents a three phase systematic approach for defeaturing. Each phase employs a different criteria for defeaturing the model. In the first phase, particular sheet metal feature is removed based on its sheet metal characteristics. The second phase uses the size of the remnant portion of a feature for deciding its candidature for removal of the feature. In the third phase all remaining negative features are temporarily removed and their stored tool bodies are brought back for piercing on the computed midsurface. Effectiveness of the defeaturing process is then computed based on the %age reduction in the number of faces during defeaturing.

The subsequent chapter presents transformation of the features of this gross shape to a generalized feature representation.

Chapter 5

Transformation of CAD Features to Generalized Feature Representation

5.1 Introduction

This chapter presents an approach for feature generalization where sheet metal feature based CAD model is transformed into generalized feature based CAD model. It begins by establishing the need for generalized features and then presents the methodology for building generalized feature based CAD model based on Spatial Grammar technique. Towards the end, this chapter demonstrates the ease with which midsurface computation can be performed on such CAD model. Following section details out the need for generalized feature based CAD model representation.

5.2 Need for Generalized Feature Representation

Feature based CAD models are widely used in development of products from different domains, such as plastics, machining, sheet metal, etc. These domains have their own feature types as per the vocabulary of the domain. For example, sheet metal domain has its own set of features such as Flange, Bend, Slots, Rip, Hem, etc.

Figure 5.1 shows a typical sheet metal part along with various features used in its construction such as Face, Flange, Bend, Cutout, etc. Figure 5.2 shows widely used sheet metal feature types. Actual number of feature types could be in the range of 60-70 for sheet metal domain only [19]. Although such diversity of features gives immense flexibility, power to the designers & modelers to build desired models, but it can be challenging for downstream applications. So, the existing feature-based midsurface computation approaches need to develop separate computation logic for each of these features [1, 67, 89]. Large number of feature types thus require huge effort in the design and implementation of midsurface computation algorithm. To avoid this problem,

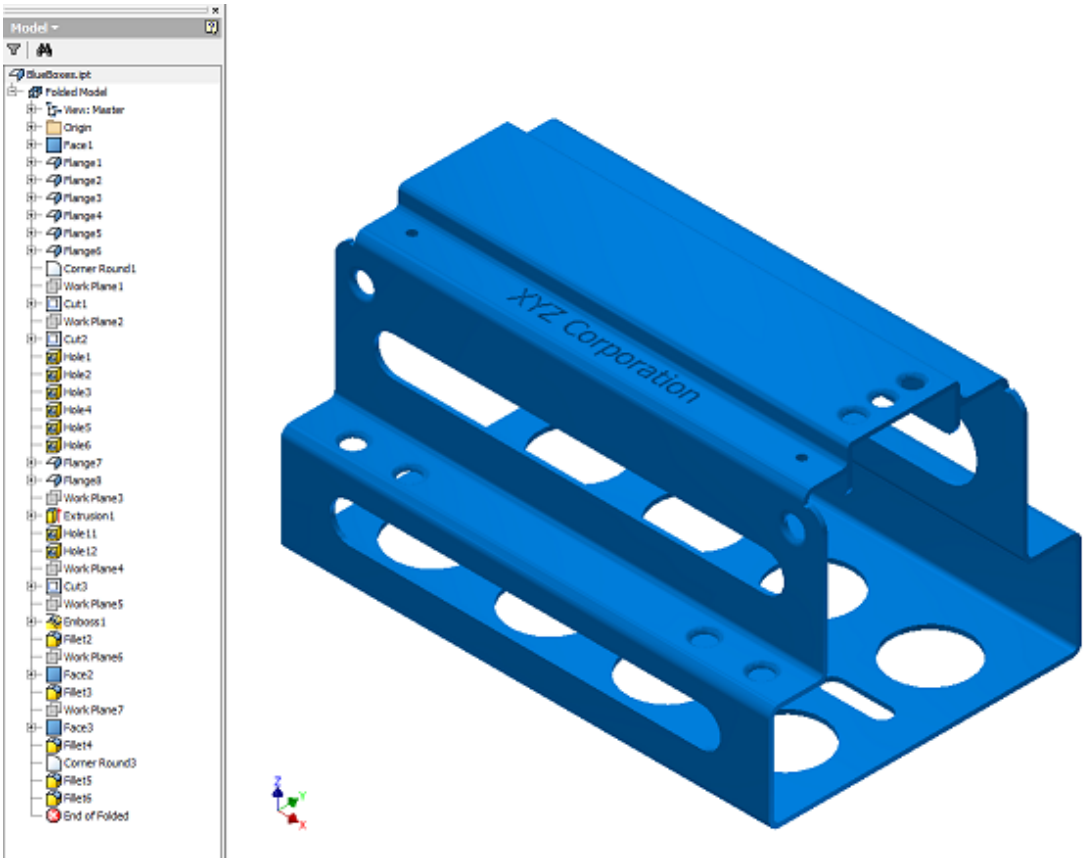


Figure 5.1: Sheet Metal CAD Model and Feature Tree

Feature type	Section of the feature
Wall	
Drawing	
Bending	
Cutout	
Hole	
Flange	
Lancing	
Coining	
Bridging	
Slot	

Figure 5.2: Sheet Metal Features

a solution has been proposed in the present research work. It has devised a generic feature form by the concept of generalization. All CAD features are transformed into a smaller set of generalized features, making it easier to write the midsurface computation algorithm.

The problem of a wide variety of features has been encountered in the past as well. Some researchers used cellular decomposed model to address this for computing midsurface [89]. The CAD model was decomposed into solid cells. The cells were subjected to feature recognition into standardized features such as Extrude [16]. Then the midsurface was computed for the Extrudes. Although these approaches work for simple models, they fail on the real-life models due to inability to recognize standard features on decomposed cells. Shapes of cells may not conform to the chosen standardized feature such as Extrude. Another problem was that, during decomposition, the original design intent would get lost which is not recovered by the feature recognition. Thus these approaches have not worked effectively in computing midsurface. The present research addresses these issues as elaborated in the following sections.

Although the reduction in variety of features helps in reducing the cases to be dealt while writing midsurface algorithm, it would further help if these small set of features themselves be represented by generalized transformations on basic geometric entities. Spatial Grammar [105], which is widely used in generative design and architecture, etc., describes construction of shapes using transformations of simple geometric entities. These entities and transformations on them are tersely represented by notations known as Spatial Grammar.

The present research uses Spatial Grammar notations to represent the generalized feature form and has been detailed below.

5.3 Related Research

Following subsections report literature on some of the domains relevant to feature generalization, such as Spatial Grammar and Feature Representation.

5.3.1 Spatial Grammar Approaches

Spatial Grammar is a general term which encompasses various shape definition grammar, like Graph Grammar, Set Grammar etc. [106]. Aim of Spatial Grammar is to bring formalism through terse but expressive definitions, validations and generation of new evolutionary shapes.

Spatial Grammar was formally introduced by Stiny and Gips [105] and is primarily used for exploratory, evolutionary design process. It is defined by set of shapes, labels and rules for geometric transformations. An example rule, say $A \rightarrow B$ first finds a shape

with label *A* and then applies the geometric transformation suggested by the rule and outputs the result *B*.

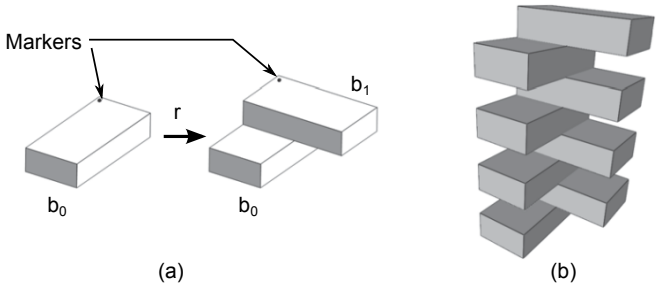


Figure 5.3: Shape Grammar Rule (Source: Hoisl [20])

Figure 5.3(1) shows a sample rule, say, r, where a newer block (b_1) is placed on the earlier block (b_0) orthogonally, after matching the marker dots. This rule, when applied successively $n = 7$ times, results in the shape shown in Figure 5.3(2). The resultant model can be represented by $n \times r(b)$. Thus a shape can be represented by an expressive and terse notation. Using variety of such rules, it is possible to build shapes. Inspite of these advantages its usage in the mainstream design applications appears marred due to complexity of defining and editing the rules.

Hoisl et al. [20] proposed a Spatial Grammar based system, with implementation in CAD. They used primitives like block, cylinder, cone as initial shapes and proposed use of Sweep feature to generate different shapes depending on different profiles and guide curves. Boolean operations were used to build more complex shapes.

CAD Grammar proposed by Deak [107] combined Shape and Graph Grammar to be more useful in Design, Modeling and Manufacturing. He claimed that traditional Shape Grammar could not work well with the CAD primitives.

Moustapha [108–110] developed interactive geometric configuration system based on Shape Grammar for architectural designs. Transformation rules were similar to language grammar rules. Although, with a wide variety of transformations, it was possible to generate different shapes, it lacked one of the key property, i.e. uniqueness. There is no process for recognizing spatially equivalent, yet, notationally different configurations.

Apart from Spatial Grammar related approaches there have been attempts to devise generalized (also known as 'neutral') feature form with which CAD models are built. This is elaborated briefly in the following subsection.

5.3.2 Feature Representation Approaches

A feature-based CAD model is built using features. Each feature adds, modifies, deletes a solid volume, thereby transforming the existing model. So, feature is a similar

entity in building a CAD model, as a rule in the Spatial Grammar. Features are specific to applications, such as sheet metal, manufacturing, etc. Due to a large variety of features, developing a generic feature-based algorithm is cumbersome. Many researchers have attempted to either devise a generalized feature form or present feature class hierarchy (taxonomy) so that similar features can be treated similarly in the algorithm.

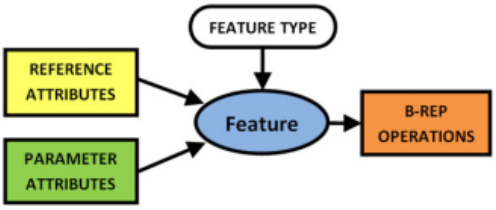


Figure 5.4: Feature Ontology (Source: Tessier [21])

In figure 5.4 Tessier showed that the generalized feature form can be represented using feature-type, solid body (B-rep) geometry and attributes.

Middleditch [111] provided abstract definition for geometric features based on cellular structure, which supported design, manufacturing applications.

Brunetti [112] combined parametric modeling with ontological reasoning for application of feature interoperability. His semantic-based shape representations was applicable to different engineering tasks using application specific ontologies.

From the literature reviewed so far it can be concluded that there has been limited number of attempts to the usage of Spatial Grammar and Feature Generalization, in CAD and its downstream applications, especially for use in feature based algorithms.

The goal of this module is to represent and transform sheet metal CAD features into generalized feature form, which is devised using a Spatial Grammar approach as elaborated below.

5.4 Proposed CAD Model Representation

Objective of this section is to present the generalized feature-based CAD model representation, based on Spatial Grammar. As the existing sheet metal feature based CAD model will be transformed in this new representation, it will need to map all the necessary entities and functionalities of any feature based CAD representation. Requirements for the proposed generalized representation are:

1. Ability to represent entities needed to build the model, such as line, circle, spline, solid, surface, etc.
2. Ability to specify transformations to spatially place the entities, such as translation, rotation, scaling, etc.

3. Ability to specify a generalized feature that can represent actual features in the model.
4. Ability to specify booleans to join or cut the feature with existing parts.

The sections below present such generalized representation in a terse Spatial Grammar like notation, to transform sheet metal feature CAD model. Midsurface algorithms are then developed based on such generalized feature based CAD model.

The proposed representation is loosely based upon Interactive Configuration Exploration (ICE) scheme developed by Hoda Moustapha [110] for generative designs in architectural domain. Although some of the fundamental entities and syntaxes are borrowed from ICE, the present research enhances it substantially to suit Mechanical CAD application and specifically, for the current context of computation of midsurface.

Following subsections give more details about the ICE scheme, then present how entities, transformations, generalized features and booleans are represented in the proposed approach.

5.4.1 Interactive Configuration Exploration (ICE) Scheme

"The ICE notation is a formalism for describing shapes and configurations, by means of their generative and relational structures" – Hoda Moustapha [110]. There are two fundamental entities in ICE, one is a *point* shown as $\bar{p}$ and another is called *Regulator*, which is an abstraction that represents a unit of action like transformation, constraints, relations, etc. A *Regulator* is represented as

$$category\mathcal{R}^{subtype,dimension}_{instance}[\{arguments\}(shapes)]$$

For example, Figure 5.5 shows the Translation regulator. Its ICE representation is shown below:

$\Delta\mathcal{A}^{T,1}[\{\bar{p}, line, n\}(shape)]$

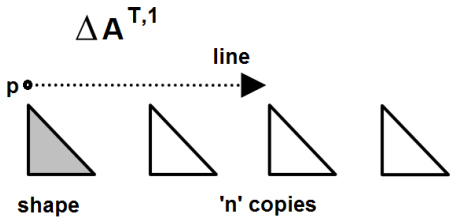


Figure 5.5: Translation of a Shape

where,

- Δ : Transformation category (type)
- $\mathcal{A}$: Affine-transformation regulator (type)
- ^{T}: subtype Translation (type)

- ^{1} : dimensionality of output (integer)
- $\bar{p}$: position (point)
- $line$: linear guide (curve)
- n : number, used for copies, scaling, etc. (float)
- $shape$: target (shape)

The example is of a triangular shape, being translated linearly, from a point p, along $line$, with n copies made. It acts as a linear-pattern operation of the triangle.

The ICE notation, can be mapped to Spatial Grammar rules typically represented as $\mathbb{A} \rightarrow \mathbb{B}$. In above example, $(shape)$ is the inputs to the rule, which can be mapped to LHS ($\mathbb{A}$) for the matching condition. On this input, transformation rule $\mathbb{A} \rightarrow \mathbb{B}$ is applied, using specified-arguments in the $\{\bar{p}, line, n\}$ to generate RHS ($\mathbb{B}$). More details of the ICE schema are in Appendix C.

The proposed representation leverages ICE notation, Spatial Grammar rules and a newly proposed feature generalization schema. The proposed generalized CAD model representation is called as $\mathcal{ABLE}$ (Affine-transformation ($\mathcal{A}$), Boolean ($\mathcal{B}$) and Loft ($\mathcal{L}$), operating on Entities($\mathcal{E}$)). It is a new paradigm to represent feature based CAD model. It consists of specifications for CAD transformations (called $\mathcal{ABLE}$ transformations – A), CAD Boolean operations (called $\mathcal{ABLE}$ Booleans – B), CAD features (called $\mathcal{ABLE}$ features – L) and CAD entities (called $\mathcal{ABLE}$ entities – E). The CAD model represented in this paradigm is called as $\mathcal{ABLE}$ model. It satisfies the requirements specified in Section 5.4 and also validates the Hypothesis 3 (Section 2.11) which states that Loft (and equivalent) feature can be used to represent rest of the sheet metal modeling features.

Following sub-sections elaborate every aspect of $\mathcal{ABLE}$ representation, i.e. Entities, Affine-transformations, Booleans and Loft. Once the representation is established, the next section presents algorithms to convert existing sheet metal features to $\mathcal{ABLE}$ Loft-equivalent features. Finally, midsurface computation of $\mathcal{ABLE}$ Loft equivalent features is elaborated.

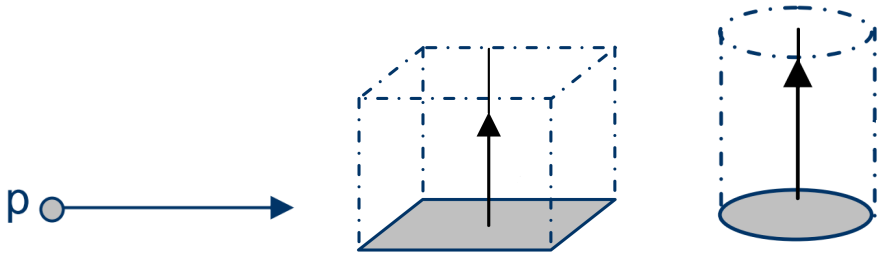
5.4.2 Proposed Representation of $\mathcal{ABLE}$ Entities ($\mathcal{E}$)

This section elaborates how various CAD entities are represented in $\mathcal{ABLE}$.

The fundamental geometric primitive in $\mathcal{ABLE}$ is a *point*. All the other geometric entities are directly or indirectly defined in terms of points. Basic tenet of $\mathcal{ABLE}$ is that most of the geometric entities and features can be represented by an operation known as Lofting (or Sweeping). In Lofting, 2D profile are lofted along a guide_curve. It is a very versatile geometric operation. Table 5.1 shows how different shapes can be built by lofting different shapes of 2D profiles along with the different shapes of *guide_curve*s.

Table 5.1: Sweeping Based Entities

Guide	line	arc	curve
2D Profile Entity			
point	line	arc	curve
line	plane	cylinder	ruled
open profile	ruled	circular	free form
rectangle	box	rect torus	rect sweep
circle	cylinder	torus	tube
closed profile	Extrude	Revolve	Sweep



(a) Line (b) Box (c) Cylinder

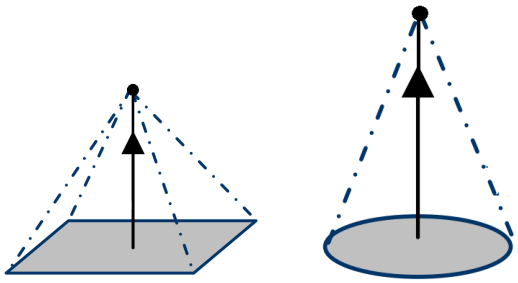
Figure 5.6: Variety of Primitives Created by Sweeping

For example, Figure 5.6 shows how Sweeping is used to create basic primitives such as a line, a box and a cylinder. Figure 5.6a shows a *point* as a 2D profile and a *line* as a *guide* when swept resulting in a *line*. Figure 5.6b shows a *rectangle* as a 2D profile and a *line* as a *guide* when swept resulting in a *box*. Figure 5.6c shows a *circle* as a 2D profile and a *line* as a *guide* when swept resulting in a *cylinder*. These results show that Sweep is a versatile operation. Lofting is more generalized than Sweeping operation. Sweeping is done only with one 2D profile and a guide curve, whereas Lofting takes multiple 2D profiles and a guide curve. So, it is not possible to create a cone by sweeping but possible by lofting. Figure 5.7 shows how pyramid and a cone can be modeled using lofting, by giving first 2D profile as *rectangle* and *circle* respectively.

Figure 5.7b shows, for cone, '2D profile 1' having a circle is lofted to '2D profile 2' having a point, along linear guide curve. Thus Lofting is a very versatile operation and it has been used as a generalized operation for defining entities and features in $\mathcal{ABLE}$.

$\mathcal{ABLE}$ introduces Object Oriented class hierarchy of geometric entities and makes *shape* as the top most base class, from which rest of the entities are derived. Advantage of such hierarchy is that once any operation is defined on the *shape* it is automatically applicable to all the entities.

Figure 5.8 shows the class hierarchy where arrow direction points from derived



(a) Pyramid (b) Cone

Figure 5.7: Variety of Primitives Created by Lofting

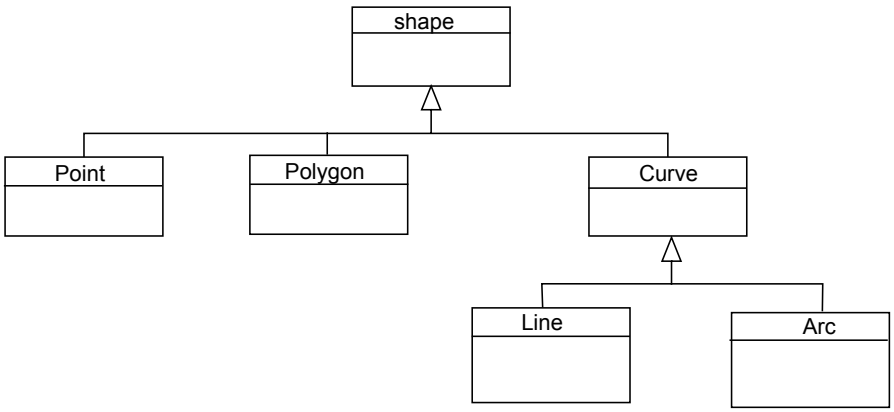


Figure 5.8: Class Hierarchy of $\mathcal{ABLE}$ Entities

(child) class to base (parent) class. Following are some of the relevant entities defined in $\mathcal{ABLE}$:

- **Shape** (*shape*): Topmost base class of Object Oriented class hierarchy for $\mathcal{ABLE}$ entities. All other entities directly or indirectly derive from it.
- **Point** (*point* :: *shape*): It is a fundamental geometric primitive expressed as $\bar{p}$. It is derived from base class *shape*.
- **Line** (*line* :: *curve*): Line is defined by two points and is derived from a generalized class called *curve*. Figure 5.9 shows a *line* defined by two points ($\bar{s_1}$ and $\bar{s_2}$). In $\mathcal{ABLE}$, it is expressed as a Loft (operation Ω of type Loft $\mathcal{L}$) of $\bar{s_1}$ along *line* with $\bar{s_1}$ as start point and $\bar{s_2}$ as end point. $\Omega\mathcal{L}^{T,1}[\{s_1, line, 0\}(\bar{s_2})^{<1>})]$

Figure 5.9: Representation of a Line

- **Arc** (*arc* :: *curve*): Figure 5.10 shows an arc defined by three points, embedding angle θ and is expressed as a Loft of $\bar{s_1}$ along circular guide with axis $\bar{t}$ and angle θ. $\Omega\mathcal{L}^{R,1}[\{s_1, \bar{t}, \theta, 0\}(\bar{s_2})^{<1>})]$

- **Circle** (*circle* :: *curve*) An *arc* with full rotation, thus having $\theta = 2\pi$ and is ex-

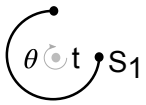


Figure 5.10: Representation of an Arc

pressed as a Loft of $\bar{s}$ along circular guide with axis $\bar{t}$ and angle 2π i.e. a full circle. $\Omega\mathcal{L}^{R,1}[\{\bar{p},\bar{t},2\pi,n\}(\bar{s})^{<0-1>})]$

- **Curve** (*curve* :: *shape*): It is a generalized curve entity modeled in terms of n points expressed as $\Omega\mathcal{L}^{C,1}[\{0,0,C_{0|1|2}\}(\bar{s})^{<1-n>})]$. It is a curve passing through n points $\bar{s}$.
- **Polygon** (*polygon* :: *shape*): It is a collection of connected lines. Figure 5.11 shows a polygon as collection ($\Pi\mathcal{C}$) of n connected *line* segments and is expressed as $\Pi\mathcal{C}^{P,1}[\{C_0\}(line)^{<1-n>})]$

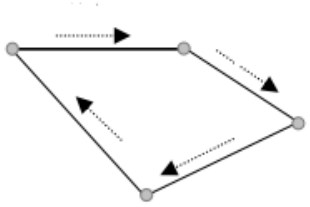


Figure 5.11: Representation of a Polygon

- **Profile** ($profile$ :: *shape*): It is a collection of n connected *curve* segments and is expressed as $\Pi\mathcal{C}^{P,1}[\{0,0,C_{0|1|2}\}(curve)^{<1-n>})]$
- **Sketch** (*sketch* :: *shape*): It is a collection of *profiles*, first outer and rest inner and is expressed as $\Pi\mathcal{C}^{S,1}[\{\}(profile)^{<1><2-n>})]$
- **Ruled Surface** ($ruledSurface$ :: *shape*): It is generated by sweeping of a curve along a line and is expressed as $\Omega\mathcal{L}^{T,2}[\{0,curve,0\}(line)]$

  Figure 5.12 shows a ruled surface modeled using collection of U,V curves and is expressed as $\Omega\mathcal{L}^{F,2}[\{0,curve^{<1-n>},C_{0|1|2}\}(curve^{<1-m>})]$

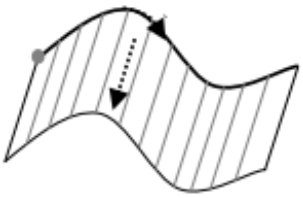


Figure 5.12: Representation of a Ruled Surface

- **Solid** (*solid* :: *shape*): It is modeled using generic surfaces and is expressed as $\Pi\mathcal{C}^{R,3}[\{0,0,C_{0|1|2}\}(surface^{<1-n>})]$

This subsection presented definitions of some of the $\mathcal{ABLE}$ entities. The next section presents how they can be transformed to position them spatially.

5.4.3 Proposed Representation of $\mathcal{ABLE}$ Affine-transformations ($\mathcal{A}$)

Affine-transformations use matrix multiplications to translate, rotate, scale entities. Such transformations are needed to place, resize the entities at appropriate location.

All these are generically clubbed together under $\mathcal{A}$ with subtypes as T, R, S for *Translation*, *Rotation* and *Scaling* respectively. The value near the sub-type $0|1|2|3$ shows that these actions can be performed on entities of various dimensions, like $point(0)$, $curves(1)$, $surface(2)$ and $solid(3)$.

- **Translation** moves *shape*, along $line$. Figure 5.13 shows translation of a triangular shape.It is expressed as $\Delta\mathcal{A}^{T,0|1|2|3}[\{0, line, 0\}(shape)]$.

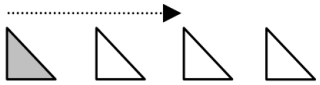


Figure 5.13: Representation of Translation

- **Rotation** rotates *shape*, along *arc* about point $\bar{p}$. Figure 5.14 shows rotation of a triangular shape.It is expressed as $\Delta\mathcal{A}^{R,0|1|2|3}[\{0, arc, 0\}(shape)]$.

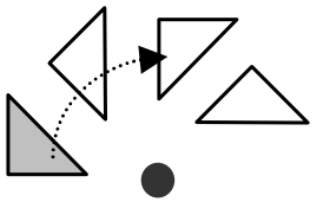


Figure 5.14: Representation of Rotation

- **Scaling** scales *shape*, by factor f, about *axis*. Figure 5.15 shows scaling of a triangular shape. It is expressed as $\Delta\mathcal{A}^{S,0|1|2|3}[\{0, axis, f\}(shape)]$.

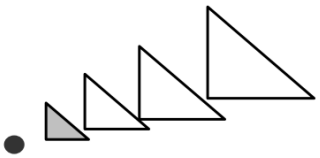


Figure 5.15: Representation of Scaling

Copy commands like *Pattern* are special cases of *Affine-transformations* with an additional parameter of *n* copies.

- **Linear Pattern** copies *shape* linearly and is expressed as $\Delta\mathcal{A}^{T,0|1|2|3}[\{0, line, n\}(shape)]$
- **Circular Pattern** copies *shape* circularly and is expressed as $\Delta\mathcal{A}^{R,0|1|2|3}[\{0, arc, n\}(shape)]$

- **Mirror** makes a single copy, mirror-ed about $\bar{t}$. Figure 5.16 shows mirroring of a triangular shape. It is expressed as $\Delta\mathcal{A}^{M,0|1|2|3}[\{\bar{p},\bar{t},0,1\}(shape)]$.

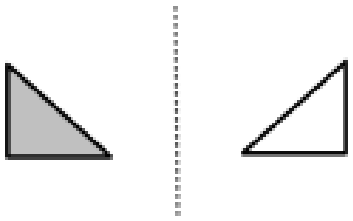


Figure 5.16: Representation of Mirroring

It is often not possible to build described geometries just by positioning pre-defined primitive geometric entities. Complex shapes are built by combines or subtracting from primitive entities. The next section presents boolean operations as defined in $\mathcal{ABLE}$.

5.4.4 Proposed Representation of $\mathcal{ABLE}$ Booleans($\mathcal{B}$)

Boolean operations are used to unite, difference or intersect two shapes, as denoted by sub-type U,D,I respectively. They are also applicable on dimensionalities like $curve(1)$, $surface(2)$ and $solid(3)$. Here, first shape, $shape_0$ is regarded as the *target body* and the result of the operation is stored in it. Rest of the shapes are termed as *tool bodies*.

- **Union** combines all the tool shapes $shape_{1-k}$ into master shape, $shape_0$. Figure 5.17 shows union of a box and a ball shape. It is expressed as $\Omega\mathcal{B}^{U,1|2|3}[\{\}(shape_{0-k})]$.

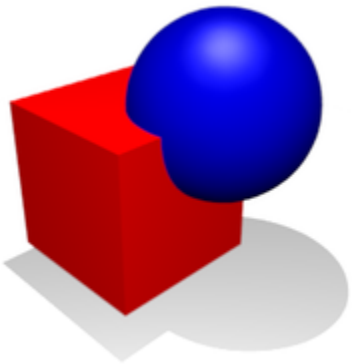


Figure 5.17: Representation of Union

- **Difference** removes combination of all the tool shapes $shape_{1-k}$ from master shape $shape_0$. Figure 5.18 shows subtraction of a ball from a box shape. It is expressed as $\Omega\mathcal{B}^{D,1|2|3}[\{\}(shape_{0-k})]$.

- **Intersection** keeps only common portion of all the shapes $shape_{0-k}$ into master shape $shape_0$. Figure 5.19 shows intersection of a ball and a box shape. It is

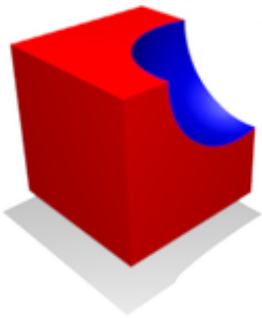


Figure 5.18: Representation of Subtraction

expressed as $\Omega\mathcal{B}^{I,1|2|3}[\{\}(shape_{0-k})]$.

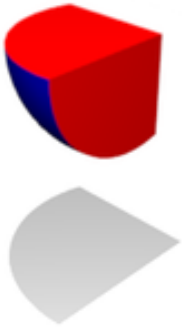


Figure 5.19: Representation of Intersection

With entities, transformations and booleans, defined so far it is possible to build CAD model, similar to CSG (Constructive Solid Geometry) representation. As mentioned in Section 5.4, the present research needs one more set of definitions, i.e. of CAD features. Following section defines generalized representation of $\mathcal{ABLE}$ CAD Features, to be used for converting sheet metal features.

5.4.5 Proposed Representation of $\mathcal{ABLE}$ CAD Features

As mentioned before, the basic tenet of the proposed model is that most of the geometric entities and features can be represented by an operation known as Lofting (or Sweeping). The Loft feature is a versatile feature which can represent other CAD features. The present research tests hypothesis 3 by demonstrating that sheet metal CAD features model can be represented as Loft feature tree, with suitable transformations and booleans. The transformed generalized CAD model tree is then sent for further steps to compute midsurface.

Loft is a generic feature capable of generating most of the other features. It joins *sketches* along a *guide_curve*.

Figure 5.20 shows a Loft having multiple sketches and a guide curve.

In $\mathcal{ABLE}$ generic Loft is represented as:

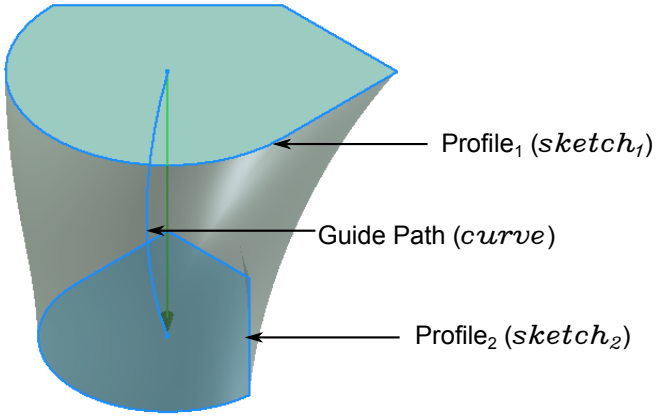


Figure 5.20: Generic Loft Feature

$\Omega\mathcal{L}^{subtype,3}[\{0,guide,0|C_{0,1,2}\}((sketch)^{<1-n>})]$

Where,

- Ω : Loft category (type)
- $\mathcal{L}$: Loft (type)
- ^{subtype}: subtypes like *R* for Revolve, *E* for Extrude, etc. (type)
- ^{3} : dimensionality of output, 3 is for solids (integer). Output of the Loft can either be *solid* (where capping faces are put to close the shape) or $surface$ (capping faces are not put) and accordingly dimensionality of $2|3$ can be specified.
- $guide$: guide curve, which can be a line, arc or any other curve
- $C_{0,1,2}$: *Continuity* options like C_0 for connectedness, C_1 for tangency and C_2 for curvature continuity can be specified at the ends where body generated by the *Loft* joins the existing shape. In case this body is disjoint or is the first one in the scene, no *continuity* is specified.
- $sketch$: sketches a Loft profiles (shape).

Loft can manifest itself in different forms as elaborated below. Figure 5.21 shows, by specific shapes of the sketches and the guide curve, it is possible to get features like Extrude, Revolve, and Sweep. These features also come with a variation known as *draft* for tapering sides. This can be modeled as *Loft* between two *sketches*, where the second *sketch* is offset-ed inside-or-outside.

Each of the features mentioned in Figure 5.21 have two variants, with and without draft. Draft is a tapering operation as explained in Figure 5.22. Figure 5.22a shows Extrude without any drafting operation, whereas Figure 5.22b shows tapering of the side faces, called “draft”.

- **Extrude**: Extrude without draft is denoted by *EnD* subtype, has single *sketch*, swept along a $line$ and is expressed as $\Omega\mathcal{L}^{EnD,3}[\{0,line,0|C_0\}((sketch)^{<1>})]$. Extrude with draft is denoted by *EwD* subtype, has two *sketches* between which loft

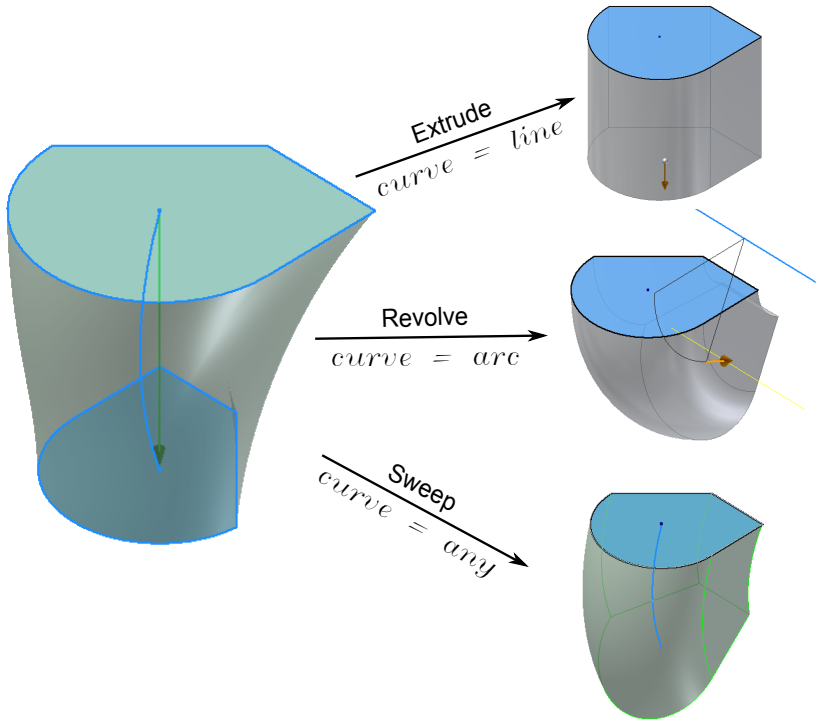
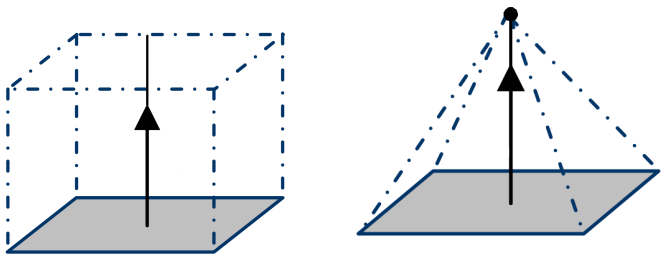


Figure 5.21: Manifestation of Loft Feature into Extrude, Revolve and Sweep



(a) Extrude without Draft (b) Extrude with Draft

Figure 5.22: Variants of Extrude

is made along a *line* and is expressed as $\Omega\mathcal{L}^{EwD,3}[\{0,line,0|C_0\}((sketch)^{<1-2>})]$.

- **Revolve**: Revolve without draft is denoted by *RnD* subtype, has single *sketch*, swept along an *arc* and is expressed as $\Omega\mathcal{L}^{RnD,3}[\{0,arc,0|C_0\}((sketch)^{<1>})]$. Revolve with draft is denoted by*RwD* subtype, has two *sketches* between which loft is made along an *arc* and is expressed as $\Omega\mathcal{L}^{RwD,3}[\{0,arc,0|C_0\}((sketch)^{<1-2>})]$.
- **Sweep**: Sweep without draft is denoted by *SnD* subtype, has single *sketch*, swept along a *curve* and is expressed as $\Omega\mathcal{L}^{SnD,3}[\{0,curve,0|C_0\}((sketch)^{<1>})]$. Sweep with draft is denoted by *SwD* subtype, has two *sketches* between which loft is made along a *curve* and is expressed as $\Omega\mathcal{L}^{SwD,3}[\{0,curve,0|C_0\}((sketch)^{<1-2>})]$.
- **Loft** is n sketches between which a loft is made along a generic *curve* and is expressed as $\Omega\mathcal{L}^{L,3}[\{0,curve,0|C_{0,1,2}\}((sketch)^{<1-n>})]$.

Primitive shapes are further specializations of $\mathcal{L}$ operators like *Extrude* and *Revolve*.

Figure 5.23 shows, by specific shapes of the sketches and the guide curve, it is possible to get primitives such as Box, Cylinder, etc.

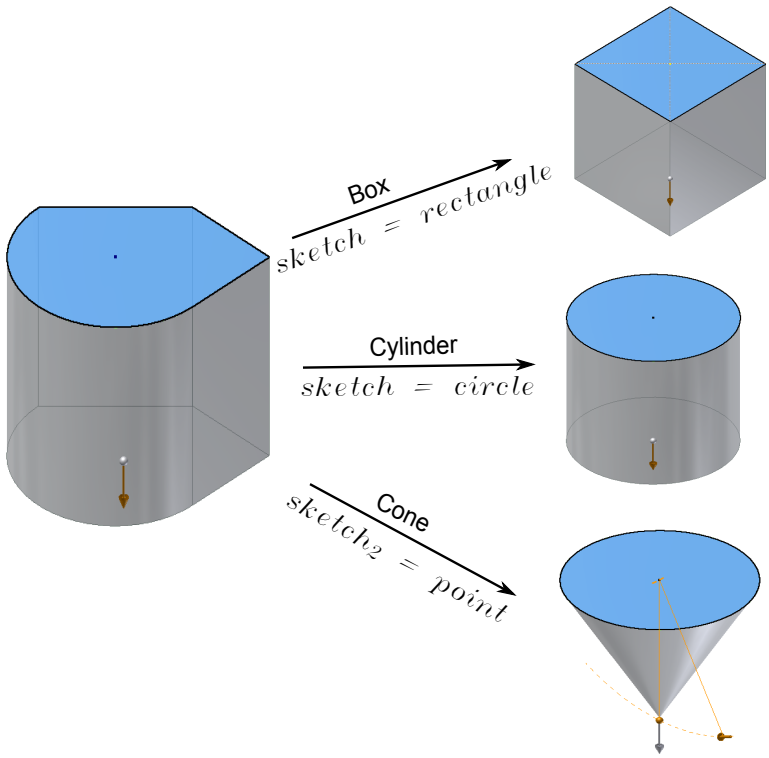


Figure 5.23: Manifestation of Extrude-Loft feature into Box, Cylinder, Cone

- **Box** : *Extrude* with *Rectangle* as sketch shape and is expressed as $\Omega\mathcal{L}^{EnD,3}[\{0,line,0|C_0\}((rectangle))]$
- **Cylinder** : *Extrude* with *Circle* as sketch shape and is expressed as $\Omega\mathcal{L}^{EnD,3}[\{0,line,0|C_0\}((circle))]$
- **Cone**: *Extrude with draft* with *Circle* as first sketch shape, point as second and is expressed as $\Omega\mathcal{L}^{EwD,3}[\{0,line,0|C_0\}((circle,point))]$
- **Torus** : *Revolve* with *Circle* as sketch shape, arc as guide and is expressed as $\Omega\mathcal{L}^{RnD,3}[\{0,arc,0|C_0\}((circle))]$
- **Sphere** : *Revolve* with *Circle* as sketch shape, point as guide and is expressed as $\Omega\mathcal{L}^{RnD,3}[\{0,point,0|C_0\}((circle))]$

Following section demonstrates how sheet metal CAD features can be modeled using $\mathcal{ABLE}$.

5.4.6 Proposed Representation of Sheet Metal Features

Similar to CAD features, sheet metal features such as Flange, Bend, etc. can also be represented by the generic $\mathcal{ABLE}$ Loft feature as demonstrated in Table 5.2. It shows

$\mathcal{ABLE}$ representations of sheet metal CAD features. The first row shows how one of the primary features, called Wall (or "Face" in Autodesk Inventor [19]), is represented in $\mathcal{ABLE}$ as an Extrude $\Omega\mathcal{L}^{EnD,3}[\{0,thickness,0|C_0\}((sketch)^{<1>})]$.

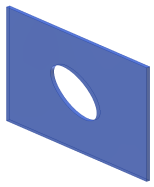
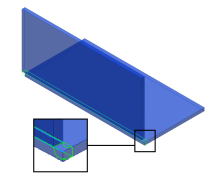
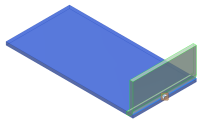
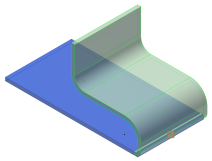
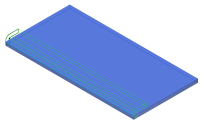
It is a Loft $\mathcal{L}$ of sub-type *EnD*, meaning Extrude with no Draft. Its extrusion direction is specified as *thickness* line. The extrusion profile is denoted by single $(<1>)$ *sketch*. The *sketch* in turn is defined as a collection ($\mathcal{C}$ of sub-type sketch *S*) of one outer profile $(<1>)$ and multiple inner profiles $(<2-n>)$. The $profile$ is represented as collection ($\mathcal{C}$ of sub-type profile *P*) of multiple $(<1-n>)$ curves. The *curve* is further defined in terms of multiple points ($\bar{s}$).

Next, Bend feature is shown as Sweep $\Omega\mathcal{L}^{SnD,3}[\{0,guide,0|C_0\}((rectangle)^{<1>})]$. It is a Loft $\mathcal{L}$ of sub-type *SnD*, meaning Sweep with no Draft. Its sweeping guide curve is specified as *guide*. The sweep profile is denoted by single $(<1>)$ *rectangle*. This *rectangle* is made up of the boundary curves of the thickness face, whose edge is selected for the bending feature. So, the *rectangle* is defined as a collection ($\mathcal{C}$ of sub-type profile *P*) of4 *lines*. The *guide* is further defined in terms of multiple points ($\bar{s}$).

Similarly, further examples present $\mathcal{ABLE}$ representations of some of the widely used sheet metal CAD features. As the present research focuses on sheet metal parts and one of the peculiarities of them is that they are of constant thickness. In this case, Sweep, rather than Loft, is the most appropriate generic form for generalization. A sweep is a special case of Loft where, instead of multiple sketches, a single sketch and a guide curve is specified. Extrude and Revolve are special cases of Sweep, so are used interchangeably as Loft-equivalent features.

Table 5.2: Loft Equivalents ($\mathcal{ABLE}$) of Some Prominent Sheet Metal features

Figure	**Feature**	$\mathcal{ABLE}$ **Features**
	**Face, Wall**	**Extrude** is created by extracting *sketch* of the "Wall" feature and giving sheet metal thickness as the distance for extrusion. $\Omega\mathcal{L}^{EnD,3}[\{0,thickness,0\|C_0\}((sketch)^{<1>})]$ Where, $sketch = \Pi\mathcal{C}^{S,1}[\{\}(profile)^{<1><2-n>})]$ $profile = \Pi\mathcal{C}^{P,1}[\{0,0,C_{0\|1\|2}\}(curve)^{<1-n>})]$ $curve = \Omega\mathcal{L}^{C,1}[\{0,0,C_{0\|1\|2}\}(\bar{s})^{<1-n>})]$

	**Cutout**	**Extrude** is created by extracting *sketch* of the "Cutout" feature and creating hole by "Subtract" boolean option. $\Omega\mathcal{L}^{EnD,3}[\{0,thickness,0\|C_0\}((sketch)^{<1>})]$ Where, $sketch = \Pi\mathcal{C}^{S,1}[\{\}(profile)^{<1><2-n>})]$ $profile = \Pi\mathcal{C}^{P,1}[\{0,0,C_{0\|1\|2}\}(curve)^{<1-n>})]$ $curve = \Omega\mathcal{L}^{C,1}[\{0,0,C_{0\|1\|2}\}(\bar{s})^{<1-n>})]$ And boolean specified as $\Omega\mathcal{B}^{D,3}[\{\}(Model,EnD)]$
	**Bend**	**Sweep** is created by creating *guide* using bend radius and a planar rectangular profile as *sketch*. $\Omega\mathcal{L}^{SnD,3}[\{0,guide,0\|C_0\}((rectangle)^{<1>})]$ Where, $rectangle = \Pi\mathcal{C}^{P,1}[\{0,0,C_0\}(line)^{<1-4>})]$ $guide = \Omega\mathcal{L}^{C,1}[\{0,0,C_{0\|1\|2}\}(\bar{s})^{<1-n>})]$
	**Flange**	**Sweep** is created by creating *guide* using bend radius, offset distance and a planar rectangular profile as *sketch*. $\Omega\mathcal{L}^{SnD,3}[\{0,guide,0\|C_0\}((rectangle)^{<1>})]$
	**Contour Flange**	**Sweep** is created by creating *guide* using contour curve and a planar profile as *sketch*. $\Omega\mathcal{L}^{SnD,3}[\{0,guide,0\|C_0\}((rectangle)^{<1>})]$
	**Hem**	**Sweep** is created by creating *guide* using hem parameters and a planar rectangular profile as *sketch*. $\Omega\mathcal{L}^{SnD,3}[\{0,guide,0\|C_0\}((rectangle)^{<1>})]$

Apart from sheet metal features, even solid modeling features may be presented as well. They are similarly transformed to their respective Loft-equivalents. Following section presents algorithms for transforming sheet metal features in the $\mathcal{ABLE}$ representations specified in Table 5.2.

5.5 Transforming Sheet Metal Features CAD Model to $\mathcal{ABLE}$ Model

Previous section explained how entities, affine-transformations, features and booleans are represented in $\mathcal{ABLE}$. This section explains the algorithm for transforming sheet metal CAD model to $\mathcal{ABLE}$ model. It traverses the input model feature tree, and converts each feature one by one to their respective $\mathcal{ABLE}$ features. When an internal or external booleans are encountered, corresponding $\mathcal{AB} : \mathcal{E}$ booleans are inserted. Similar transformations and entities are transformed as well.

Algorithm 4 Transforming Input CAD Model to $\mathcal{ABLE}$ Model

Require: A Sheet Metal FCAD (*model*) with access to the feature tree

1: **while** $model \rightarrow nextFeature() != null$ **do**
2: $f_i = currentFeature()$
3: $af_i \rightarrow converTo\_ABLE\_feature()$
4: $list \rightarrow add(f_i)$
5: $model \rightarrow add(af_i)$
6: **end while**
7: $model \rightarrow remove(list)$
8: $model \rightarrow rebuild()$

Algorithm 4 presents the overall steps of transformation of input sheet metal features CAD model to $\mathcal{ABLE}$ CAD features i.e. Loft-equivalent features.

1. The model feature tree is traversed (Algorithm 4 lines:1-6 loop).
2. The current feature is converted to its Loft-equivalent (Algorithm 4 lines: 2-3).
3. The existing feature is added to a list to be removed later (Algorithm 4 line: 4).
4. The converted feature is added to the model (Algorithm 4 line: 5).
5. The features from the old-features-list are removed (Algorithm 4 line: 7).
6. The model is rebuilt (Algorithm 4 line: 8).

Following paragraph explains how the function $converTo\_ABLE\_feature()$ (Algorithm 4 lines: 3) works for each input CAD feature by taking an example input model shown in Figure 5.24 and transforming it to $\mathcal{ABLE}$ model as shown in Figure 5.27.

1. The model feature tree which has 5 features, is traversed one feature at a time (Algorithm 4 lines:1-6 loop).

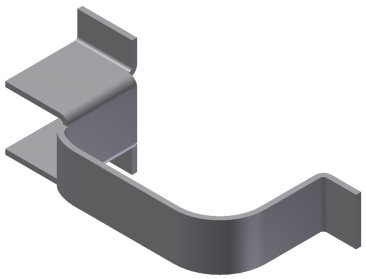
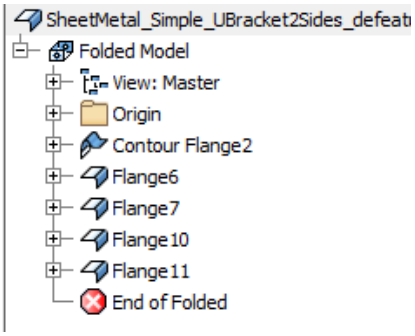
Input CAD Model	**Tree**
	

Figure 5.24: CAD Model Built with Sheet Metal Features

2. The first feature encountered is "Contour Flange2". Its parameters are Edge and Contour Curve. Its conversion can be done in one of the two ways, as follows:

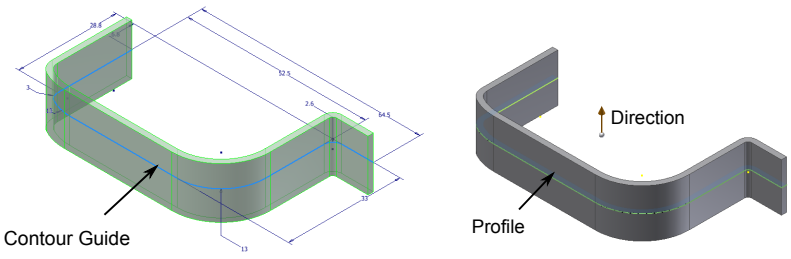
   (a) Extrude $\Omega\mathcal{L}^{EnD,3}[\{0, distance, 0|C_0\}((sketch)^{<1>})]$ is computed as follows:

Algorithm 5 Contour Flange to $\mathcal{ABLE}$ Extrude

Require: Contour Curve, Edge

1: While in rollback state, extract Contour Curve from the feature and find length of the Edge and distance.
2: Sketch creation method 1: Offset the curves and add capping lines to make closed sketch. Create a new sketch and copy curves in.
3: Sketch creation method 2: Extract boundary curves of the face having Contour Curve. Create a new sketch and copy curves in.
4: Extrude the sketch with the distance.

Figure 5.25 shows input feature parameters of Contour Flange and its transformation to equivalent Extrude feature. Algorithm 5 details to steps.



(a) Contour Flange Parameters (b) Extrusion' Parameters

Figure 5.25: Contour Flange to Extrusion

   (b) Sweep $\Omega\mathcal{L}^{SnD,3}[\{0, guide, 0|C_0\}((rectangle)^{<1>})]$ is computed as follows:

Algorithm 6 Contour Flange to $\mathcal{ABLE}$ Sweep

Require: Contour Curve, Edge

1: While in rollback state, extract rectangle from thickness Face adjacent to the Edge and the Contour Curve.
2: Sketch creation method: Extract boundary curves of the face. Create a new sketch and copy curves in.
3: SWEEP the sketch along guide

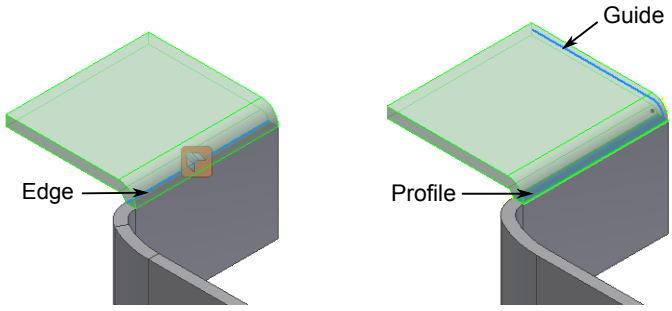
(c) Boolean: As this is the first feature, Boolean is not specified.

(d) Output $\mathcal{ABLE}$ model built so far is:

+ $Extrusion2 = \Omega\mathcal{L}^{EnD,3}[\{0, edge1, 0|C_0\}((sketch1)^{<1>})]$

3. The second feature encountered is "Flange6". Its parameters are Edge, Flange Distance and Offset distance. Its conversion can be done as follows:

(a) Sweep $\Omega\mathcal{L}^{SnD,3}[\{0, guide, 0|C_0\}((rectangle)^{<1>})]$ is computed as follows:



(a) Flange Parameters (b) Sweep Parameters

Figure 5.26: Flange to Sweep

Figure 5.26 shows input feature parameters of Flange and its transformation to equivalent Sweep feature. Algorithm 7 details to steps.

Algorithm 7 Flange to $\mathcal{ABLE}$ Sweep

Require: Edge, Flange Distance and Offset distance

1: While in rollback state, extract rectangle from thickness Face adjacent to the Edge.
2: Sketch creation method: Extract boundary curves of the face. Create a new sketch and copy curves in.
3: Guide creation method: Compute 3 connected curves using Offset Distance, Bend Radius and Flange Distance.
4: SWEEP the sketch along guide

(b) Boolean: *Union* between the second feature with the existing first feature (shown as *model* so far) and is denoted as: $\Omega\mathcal{B}^{U,3}[\{\}(Extrusion2, Sweep1)]$

(c) Output $\mathcal{ABLE}$ model built so far is:

+ $Extrusion2 = \Omega\mathcal{L}^{EnD,3}[\{0, edge1, 0|C_0\}((sketch1)^{<1>})]$
+ $Sweep1 = \Omega\mathcal{L}^{SnD,3}[\{0, guide, 0|C_0\}((rectangle)^{<1>})]$, $\Omega\mathcal{B}^{U,3}[\{\}(model, Sweep1)]$

4. Similar transformation happens for the remaining features and the Output $\mathcal{ABLE}$ model built looks like (Figure 5.27), and is shown as follows:

+ $Extrusion2 = \Omega\mathcal{L}^{EnD,3}[\{0, edge1, 0|C_0\}((sketch1)^{<1>})]$
+ $Sweep1 = \Omega\mathcal{L}^{SnD,3}[\{0, guide1, 0|C_0\}((sketch2)^{<1>})]$, $\Omega\mathcal{B}^{U,3}[\{\}(model, Sweep1)]$
+ $Sweep2 = \Omega\mathcal{L}^{SnD,3}[\{0, guide2, 0|C_0\}((sketch3)^{<1>})]$, $\Omega\mathcal{B}^{U,3}[\{\}(model, Sweep2)]$
+ $Sweep3 = \Omega\mathcal{L}^{SnD,3}[\{0, guide3, 0|C_0\}((sketch4)^{<1>})]$, $\Omega\mathcal{B}^{U,3}[\{\}(model, Sweep3)]$
+ $Sweep4 = \Omega\mathcal{L}^{SnD,3}[\{0, guide4, 0|C_0\}((sketch5)^{<1>})]$, $\Omega\mathcal{B}^{U,3}[\{\}(model, Sweep4)]$

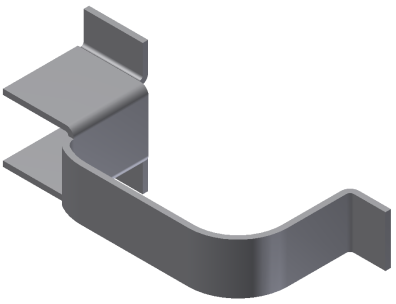
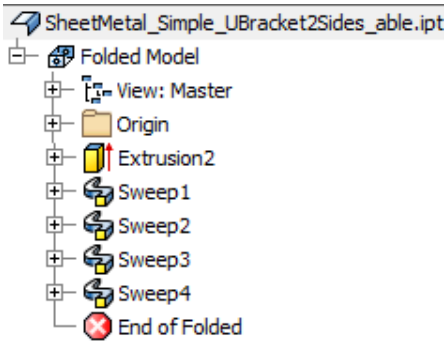
Output $\mathcal{ABLE}$ based Model	**Tree**
	

Figure 5.27: Output $\mathcal{ABLE}$ Based Model

The output shows how the use of Spatial Grammar has made it possible to come up with a representation (as demonstrated by $\mathcal{ABLE}$) so that algorithms need not have to work with all sheet metal features but just a few generalized features. Following section demonstrates how the generalized $\mathcal{ABLE}$ features compute the midsurface.

5.6 Significance of $\mathcal{ABLE}$ Paradigm

This chapter elaborates the proposed $\mathcal{ABLE}$ representation implemented in **MidAS** system. Significance of using $\mathcal{ABLE}$ based feature-definition is enumerated below:

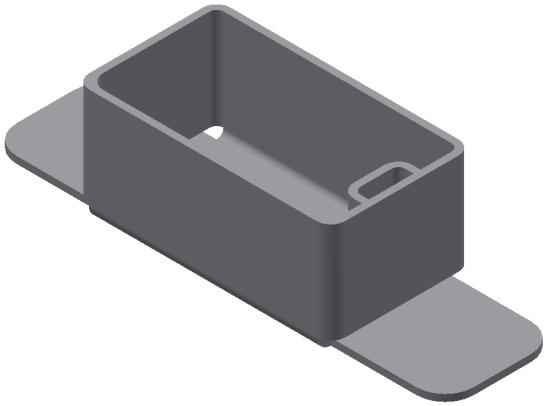
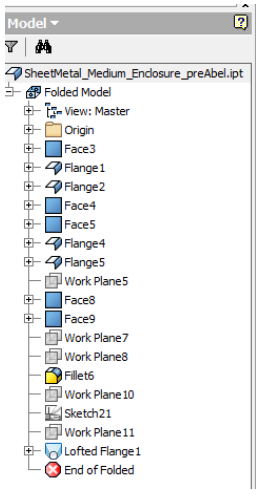
1. Spatial Grammar, which has been largely used in exploratory, generative and architectural applications, has been used in the present research to represented mechanical engineering CAD features in a generic way so that they can be applicable in various CAD systems.
2. Features which are similar in nature can be put in a single category in $\mathcal{ABLE}$. All sketch based features, such as, Extrude, Revolve, Sweep, Loft can be under *SketchFeature* category and then individual types can be the 'subtype' field.
3. In case of feature based modeling, although variation in parameters is possible, changing the shape dramatically is not possible (to an extent it is possible with Replace Face kind of features), but Shape Grammars are very generic and can generate wide range of shapes. Feature-definition-constraint rules, should be imposed to restrict allowable changes only.

Following section elaborates generation of $\mathcal{ABLE}$ model for a practical part model.

5.7 Example

This section shows generalization on the defeatured "Encloser" model. It has Sheet metal features such as Face, Flange, Lofted Flange etc.

Table 5.4: Generalizing "Enclosure" Model

Model	Feature Tree
	

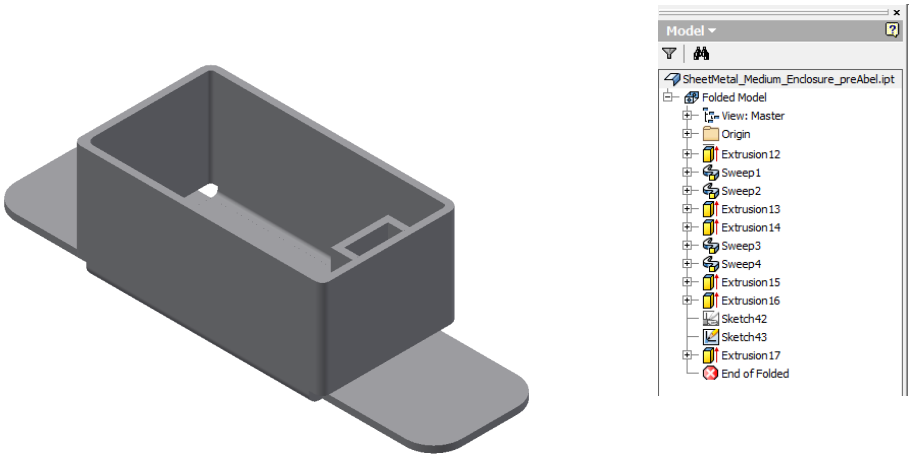


Table 5.4 shows transformation of sheet metal features CAD model into $\mathcal{ABLE}$ model having only Loft-equivalent features. This result demonstrates the effectiveness of generalization module as the input model is faithfully transformed into generalized features without any feature or shape loss.

5.8 Conclusions

This chapter has proposed a new CAD model representation paradigm called $\mathcal{ABLE}$, to provide generalized definition of CAD features, as well as for application specific features like sheet metal features. Using just a few entities such as Loft($\mathcal{L}$), Booleans($\mathcal{B}$) and Affine-transformations($\mathcal{A}$) along with entities, a CAD model can be defined adequately. This chapter also presents how a feature based algorithm, e.g. the midsurface-computation, can be effectively devised using the generalized definitions provided by $\mathcal{ABLE}$.

Chapter 6

Generation of Midsurface from Generalized Sheet Metal CAD Part Model

6.1 Introduction

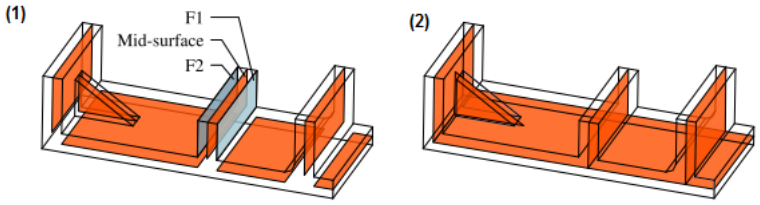
This chapter presents algorithms for generating a quality midsurface from a generalized feature-based CAD model.

Despite a wide demand for midsurface, its quality has not improved over time. The literature survey findings, quoted in Chapter 2, show that the existing approaches fail to compute a well-connected midsurface, especially in the case of complex models. One of the widely used approaches of midsurface generation, the Face Pairing approach, has several limitations such as difficulties in detecting face pairs for computing midsurface patches and devising a generic approach for joining these patches. Typical failures are, missing midsurfaces, gaps, overlaps, midsurfaces not lying midway, etc. Correcting these errors is mostly a manual, laborious and time-consuming process, requiring from several hours to days. This chapter proposes use of feature based cellular decomposition of the generalized feature based CAD model to address these issues which are explained in the sections to follow.

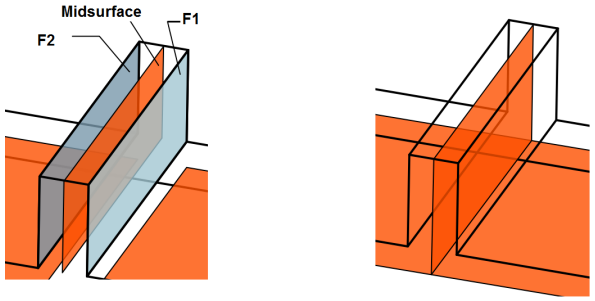
6.2 Limitations of Existing Approaches

Review of the past approaches indicate that, Face Pairing (also known as Midsurface Abstraction) is one of the most representative approaches for midsurface generation [10]. In the Face Pairing approach, faces opposite each other are identified by ray-firing method and are grouped as face-pairs [36].

Figure 6.1 shows an example model in two stages of midsurface computation. Fig-



(a) Full model view



(b) Magnified view

Figure 6.1: Face Pairing Approach (Source: Boussuge [14])

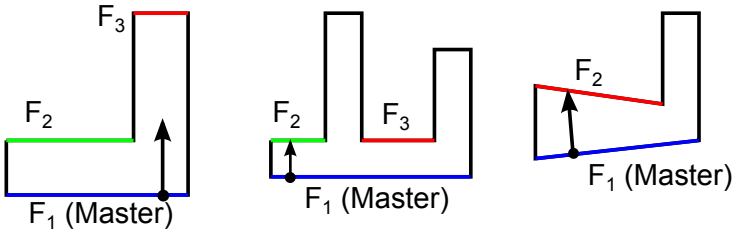
ure 6.1a(1) shows midsurface patches whereas Figure 6.1a(2) shows them joined. Figure 6.1b shows magnified view of the same model. In Figure 6.1a(1), one of the face-pairs is marked for an example, as $F_1 - F_2$. The face from which a ray is fired is called as "master face" (F_1) and the target which got identified as the opposite face, is called as the "slave face" (F_2). Figure 6.1a(2) shows the patches extended or trimmed, brought at a common edge and then joined to form a well-connected midsurface. The face pairing approach scores over the other approaches such as MAT, as it is intuitive and does not need post-processing to remove extra branches [36]. It is widely researched and used in academic as well as commercial CAD-CAE systems.

In-spite of its wide usage, the face pairing approach however has two critical issues, namely, Face Pairs Detection and Midsurface Patch Joining. These issues aggravate as the complexity of the model increases and result in various errors in the output midsuface. These are explained in the following subsections.

6.2.1 Face Pairs Detection Problem

For real-life complex CAD models detecting opposite faces is challenging [7, 113]. Some of such scenarios are described below:

- **Missing target**: If a face is not within threshold distance, or not in the direction of the ray-fire from the master, it fails to get detected as the slave. In Figure 6.2a, F_1 is the master face from which a ray is fired (shown by an arrow). It does not hit F_2, so it does not get qualified as the slave face. The ray hits F_3 but it is too far way, so fails the threshold distance criteria, thus F_3 also does not get qualified as



(a) Missing target (b) Multiple targets (c) Angled targets

Figure 6.2: Problems in Identification of Face Pairs

the slave. So, incorrect choice of the location from which ray is fired, misses as the otherwise qualified slave face, such as F_2 in this case.

- **Multiple targets**: There could be multiple candidate salve faces for a single master face, approximately at similar distances, but, either due to tolerance issues or due to location from which the ray is fired, only one of them gets selected, discarding other equally eligible candidate slave faces. In Figure 6.2b, master face F_1 correctly identifies F_2 but does not find F_3 as the ray was not fired from the location which would find it. So F_3 does not get added to the same face pair as with F_2. Ideally a face pair with F_1 as master face and F_2&F_3 as slave faces should have got formed.
- **Angled targets**: In case of variable thickness shapes, distance between master and slave faces vary, making it challenging to decide appropriate slave face. Figure 6.2c shows that F_1 is not able to pair up with F_2 due to being farther away from the allowed distance threshold.
- **Non-planar targets**: Non planar geometries such as free-form surfaces also pose problems in calculation of distances and finding slave faces [7].

Such undetected or inappropriately detected face pairs do not create appropriate midsurface patches thus leaving gaps in the overall midsurface output. Next section discusses problems encountered in joining the midsurface patches.

6.2.2 Midsurface Patches Joining Problems

Midsurface patches are generated from each face pair, either by offsetting the master face in the middle or by computing a surface equidistant from both, master and slave faces. As seen in Figure 6.1b, they are then joined, either by extending or trimming up-to a common edge between them. Failing to compute correct extensions, trimmings, etc. result in unconnected midsurface patches.

Figure 6.3 shows cases where the midsurface patches, either got extended too much or failed short of the common edge.

Basic principle of joining the midsurface patches is that they need to be connected

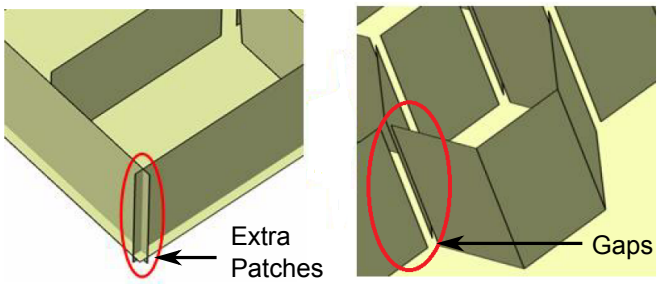
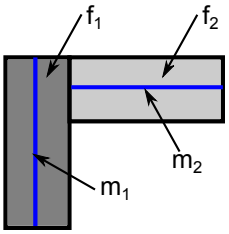
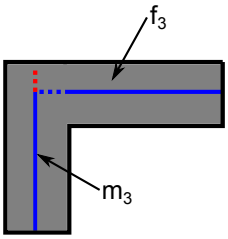
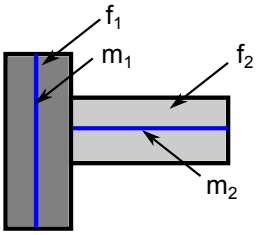
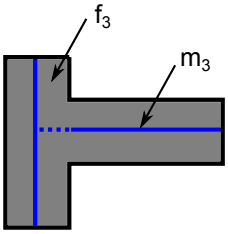
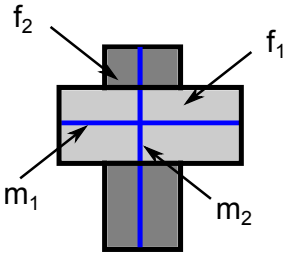
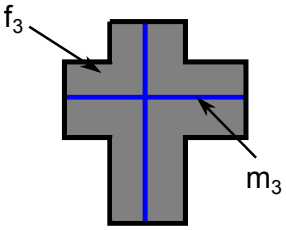


Figure 6.3: Problems in Joining Midsurface Patches (Source: Sheen [6])

in the same manner as their corresponding face-pairs are connected, in the input model. Detecting when to extend the midsurface patches and by what amount, also, when to trim the midsurface patches and by what amount, are complex problems. These depend on various configurations in which face pairs or features (and their corresponding midsurface patches) are connected in the input model.

Table 6.1: Variety of Configurations of Interactions of Midsurfaces Patches

Configuration	**Interaction**	**Midsurface**	**Midsurface Joining Strategy**
L			m_2 needs to be extended to meet m_1, whereas m_1 needs to be trimmed.
T			m_2 needs to be extended to meet m_1, whereas m_1 remains unchanged.
X			Both m_1 and m_2 remain unchanged.

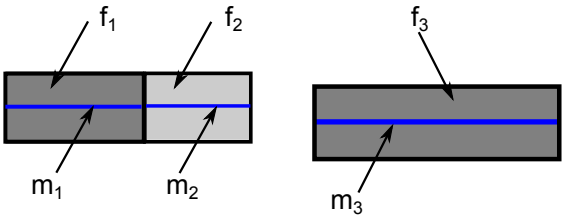
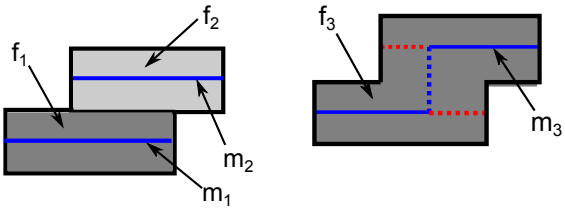
Align		Both m_1 and m_2 remain unchanged.
Overlap		The extra patches are removed from both m_1 and m_2, whereas both are joined with an additional patch.

Table 6.1 illustrates some of the representative configurations of interaction of two face pairs (or features), along with their corresponding midsurface patches. Face pairs (or features) f_1 and f_2 include m_1 and m_2 as their corresponding midsurface patches. Both interact in multiple ways, which can be commonly referred to as 'L', 'T', 'X', Align and Overlap configurations. Each configuration needs different adjustments to midsurface patches to get them connected.

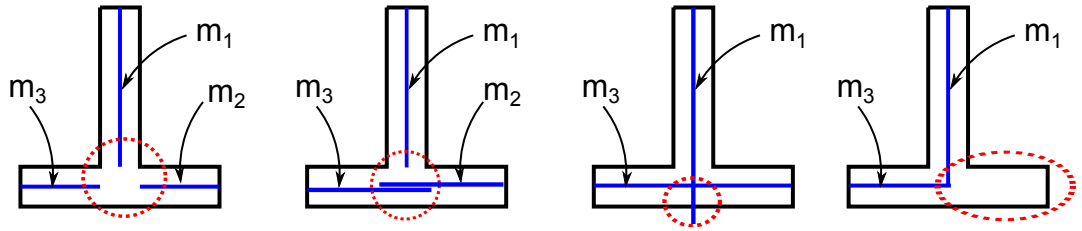
The first row of Table 6.1 shows the configuration type 'L'. The first picture depicts interaction of face pairs f1 and f2 and the second picture shows adjustments needed to connect two midsurface patches m_1 and m_2 i.e. m_2 needs to be extended to meet m_1, whereas m_1 needs to be trimmed.

The second row of Table 6.1 shows the configuration type 'T', where two midsurface patches get connected by logic: m_2 needs to be extended to meet m_1, whereas m_1 remains unchanged.

Similarly further rows elaborate remaining configuration types. The table clearly shows that developing the midsurface patch joining logic involves dealing with a variety of different configurations, which need to be addressed on case-by-case basis. This makes devising a generic midsurface patch joining approach, a challenging task. Thus failures to join midsurface patches are very common and manifest in the form of gaps, overlaps, etc.

Figure 6.4 shows some of the commonly occurring midsurface errors. Figure 6.4a shows that midsurface patches (m_1, m_2, m_3) could not extend properly thus resulting in gaps between them. Figure 6.4b shows that two of the patches (m_2, m_3) got extended more than necessary thus resulting in overlaps. Figure 6.4c shows that one patch (m_1) got extended so much that the midsurface went out of the model's boundaries. Figure 6.4d shows that one of the patches (m_2, not shown) did not get created at all.

To get the idea of gravity of this challenging problem, Figure 2.20 shows the output



(a) Gaps (b) Overlaps (c) Extra Extension (d) Missing Patches

Figure 6.4: Typical Errors in Generating Connected Midsurfaces

of automatic midsurfacing by two of the leading CAD-CAE commercial systems, which primarily use face Pairing approach.

CAD models of English alphabets are chosen for benchmarking as they are easy to understand and represent a wide variety of shapes and interaction types found in the real-life thin-walled parts. Figures 2.20b and 2.20c demonstrate the midsurface failures such as missing surfaces, gaps, not lying midway, etc.

Developing a generic logic for joining midsurface patches is thus the need of the hour. As there is no theoretical framework encompassing all the possible interaction types, so far an unified logic has not been possible [15].

The present research leverages generalized feature-based CAD model to address the issues related to face pairing. It proposes to use cellular decomposition approach to address the problems of midsurface patch joining.

6.2.3 Using Feature Information Instead of Face Pairing

As seen in Section 6.2.1 detecting face pairs is a challenging and error prone task. Information stored in the features in the feature-based CAD model has potential of replacing the need of face pairing altogether. In feature-based CAD modeling paradigm, model is built step-by-step using features. At each step, a new feature is booleaned to (i.e.'added to' or 'subtracted from') the existing model, thus making final model at the end.

The present research proposes to use each feature's parameters to compute its own midsurface patch and thus avoiding the need of face pairing altogether. For example, Extrude feature computes its own midsurface using own sketch-profile and other parameters. This idea is advantageous, because, it not only avoids the problems of face pairing but it is also regenerates models upon parameter changes. Whenever a feature undergoes any dimension change, only the dependent features are updated to maintain the validity and constraints defined in the model. Thus, in the idea of computing midsurface from feature parameters, only the features which have updated need to re-compute the midsurface patches, whereas in the traditional face pairing approach, the whole model needs to be reevaluated for face pair detections and midsurface computations.

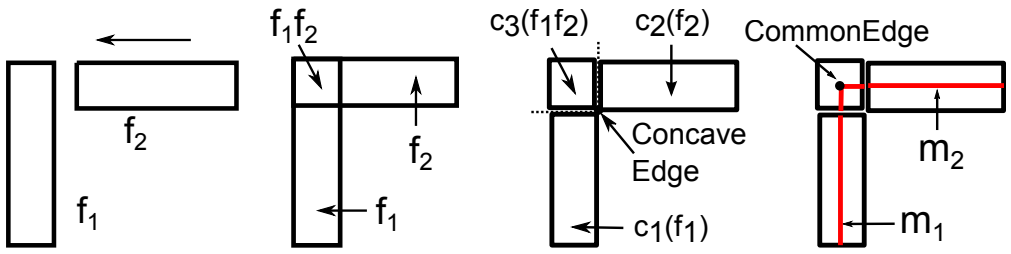
One of the limitations of using features in midsurface computation is the variety of features present in a typical sheet metal parts CAD model. All these features would need to have their own logic of computing midsurface patches. Another limitation is that even after implementation is over with the available set of features, introduction of any new feature would need additional logic for computing their own midsurface patches. These problems can be avoided by proposing use of generalized features, i.e. $\mathcal{ABLE}$ model paradigm. As elaborated in Chapter 5, $\mathcal{ABLE}$ models are made up of only a set of loft-equivalent features such as Extrude, Revolve, Sweep and Loft. With $\mathcal{ABLE}$ model as input, the midsurface patch generation needs to be implemented only for limited number of loft-equivalent features.

Thus the proposed solution of using $\mathcal{ABLE}$ model avoids face pairing altogether, making midsurface patch generation less error prone.

6.2.4 Using Feature-based Cellular Topology for Joining Patches

In Section 6.2.2, Table 6.1 showed a variety of configurations in which midsurface patch joining takes place in the existing face pairing approaches. It can be clearly seen that there are a variety of configurations that need to be solved for connecting midsurface patches. The reported past approaches have not demonstrated any generic approach to solve them generically, i.e. with a singular rule.

The present research uses feature-based cellular topology (FBCT) to address the problems in joining midsurface patches. Figure 6.5 shows a schematic representation of a typical feature interaction, explains the context of feature-based cellular decomposition (FBCD), its outcome i.e. FBCT and its advantage in devising generic midsurface patch joining approach.



(a) Features (b) Boolean (c) FBCD (d) FBCT with Midsurface

Figure 6.5: Feature Interactions and Cellular Decomposition

Figure 6.5a shows feature f_2 getting booleaned to the existing model, denoted by feature f_1. After boolean, both features are merged with some portion (shown as $f_1 f_2$) of f_2 getting consumed inside f_1, as seen in Figure 6.5b. Figure 6.5c depicts the cellular decomposition with cells having owner features such as $c_1(f_1)$, $c_2(f_2)$ and $c_3(f_1 f_2)$. Figure 6.5d shows same cells with midsurface patches computed for each. Table 6.1

clearly indicates that $f_1 f_2$ portion is the place where midsurface patches interact differently in different configurations.

Cellular decomposition refers to technique of decomposition of a shape into subshapes called cells. Primary advantage of cellular decomposition is simplification. Instead of working on complex shapes, cells having primitive shapes, are easier to handle. For example, instead of meshing whole model, cells are meshed based on their characteristics. Cellular decomposition not only reduces the number of types of shapes to be handled, but also creates a possibility of working on them parallelly. Because of such substantial advantages, cellular decomposition has been used in variety of domains such as CAM for pocket machining, CAE for isolating sub-volumes with different meshing patterns, CAD model transitions with different level of details (LoDs), etc. Cellular Decomposition suffers from certain limitations as well. The choice of locations and cutting-geometries for decomposition decides the quality of resultant decomposition. Wrong choices result in a large number of redundant cells or in the cells which are not of primitive shapes.

6.3 Proposed Approach for Midsurface Computation

As discussed in the previous sections, two of the most critical problems faced in the face pairing approach are, face pair detection and midsurface patch joining. The present research proposes to leverage generalized Loft-equivalent features to address the issues related to face pairing and use feature-based cellular topology to address the problems of midsurface patch joining. The proposed solution is to separate out the common area by decomposition, so that generic rules for joining midsurface patches, can be derived.

The generalized feature-based CAD model is suitably decomposed into manageable cell volumes. Cells are classified into solid cells (*sCell*) and interface cells (*iCell*). Midsurface patches are first computed from *sCell*s and are then joined in the *iCell*s. The detailed treatment is presented in the sections to follow. To begin with Cellular Decomposition approaches are reviewed in the below section.

6.3.1 Literature Review of Cellular Decomposition

Cellular decomposing has been researched extensively, especially for CAM applications [114]. Manufacturing sequences need different cells to be assigned with different machining processes. Generation of such cells is done by cellular decomposition of the CAD model such as Brep model. Similarly, in CAE, identification and separation of cells is carried out to assign different meshing patterns to different cells. Following are some of the relevant past approaches for cellular decomposition:

Bih-Yaw Shih [115] presented an automated swept volume detection and decompo-

sition algorithm for hexahedral mesh generation. This approach was computationally intensive for complex model.

Yong Lu [116] reported the work on shape recognition and volume decomposition to automatically decompose a CAD model into hex meshable volumes.

Yoonhwan Woo [22, 55, 92, 117, 118] did extensive work in fast and maximal cellular decomposition. He found that in typical cell decomposition, each of the faces having a concave edge, act as cutting geometries and are extended spanning whole model, generating a large number of cells. Many of them were redundant. To avoid this he proposed a concept of 'maximal volume'.

Figure 6.6 depicts Woo's process in which (a) shows the input model and (b) shows result of the cellular decomposition, generating large number of cells. His process combines cells based on certain criteria. Figure 6.6c shows the output having far lesser number of cells, i.e. just 2 cells having primitive shapes.

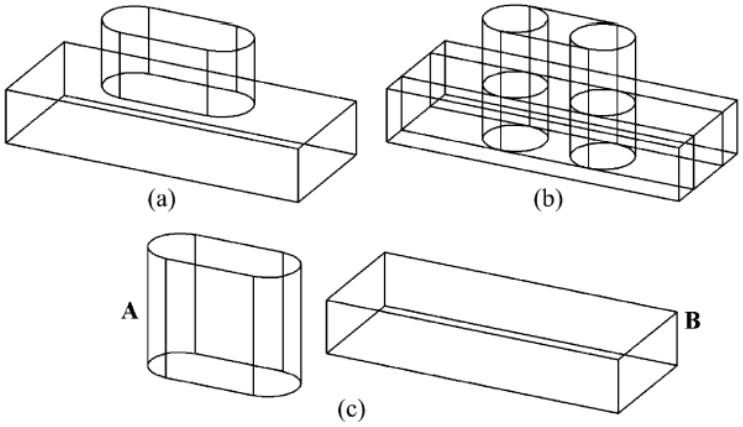


Figure 6.6: Maximal Cellular Decomposition (Source : Woo [22])

Woo's method was superior to other prevalent methods but was restricted to shapes with analytical surfaces only.

The result of Cellular Decomposition is a collection of cells, termed as 'cellular topology (CT)' model [119]. Cellular decomposition of a feature-based CAD model results in feature-based cellular topology (FBCT) model. Figure 6.7 shows one such example of FBCT model, in which, $cell1$ has $block$ as the owner feature whereas $cell2$ is the common cell for features $block$ and $roundedRectPocket$, etc.

Cellular Topology (CT) is the solid modeling data structure having cells as the fundamental building block. Cells do not overlap volumetrically but touch each other only at surface boundaries. In Feature-based cellular topology (FBCT) cells have owner features. FBCT model has not yet been used widely, especially in commercial systems. Some of the relevant works in academic research are reviewed below.

Bidarra [120–122] proposed feature based cellular topology to improve over the

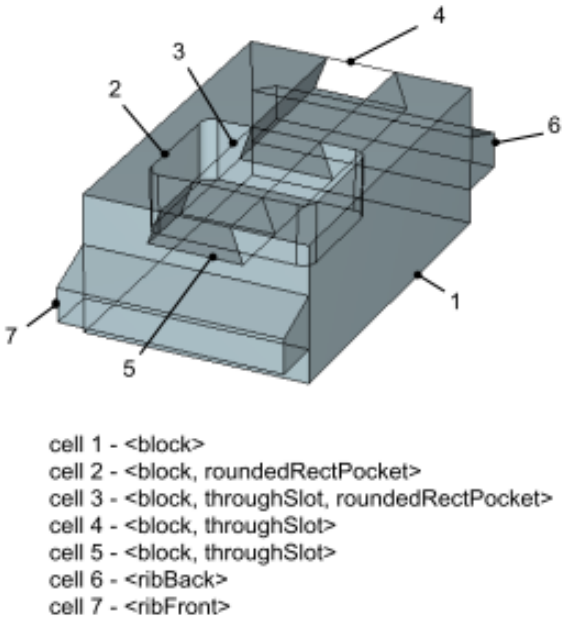
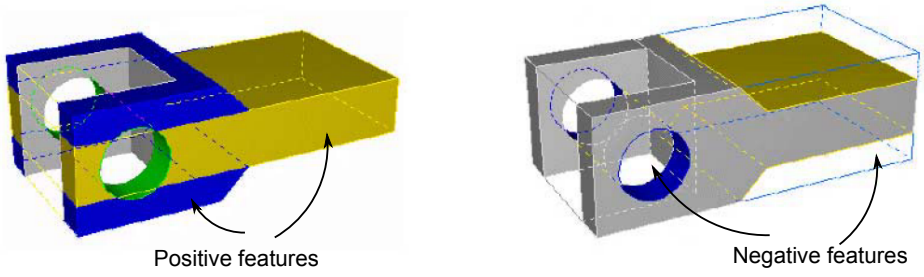


Figure 6.7: Feature-based Cellular Topology (Source : Bidarra [23])

shortcomings of B-prep, such as complex boundary evaluation after any modeling operation. He built a FBCT system called *SPIFF* in which features were assigned to cells of ACIS CT model. The system provided flexibility to choose specific cells for evaluating boundary for specific applications.



(a) Design view **(b) Manufacturing view**

Figure 6.8: Application Perspective Based Viewing of a Model (Source: Van [24])

Kraker [123] presented feature operations in cellular model for multi-view architecture. This architecture was used to selectively present cells as per need of client application. For example, a design can see both additive and subtractive features, whereas manufacturing view of the same model can see only subtractive features. Figure 6.8 shows these two views with an example. Use of positive and negative features is different in both the views, even though the model is same.

Review of the past approaches show their wide applicability for the processes like machining, meshing, etc. There are a large number of approaches developed to cater to various domains. The present research suitably leverages one of the feature-based

cellular decomposition approach, a combination of Woo [22, 55, 92, 117, 118] and Bidarra [120–122] approaches. As a readily usable implementation of such combination is not available, a manual cellular decomposition approach has been taken up, based on the methodology suggested in the said approaches. Following section elaborates the approach taken in the present research.

6.3.2 Proposed Feature-based Cellular Decomposition Approach

The primary objective of this module is to present an approach taken for decomposing a generalized feature based CAD model. Following steps elaborate the approach:

- **Feature separation**: A feature-based CAD model is built by Boolean of new features with the existing. Booleans are of "Unite", "Subtract", "Intersect" and "New Body' types. In this step, the "Unite" boolean type set between two features is removed and instead "New Body" type applied, thereby separating the tool bodies of the features.

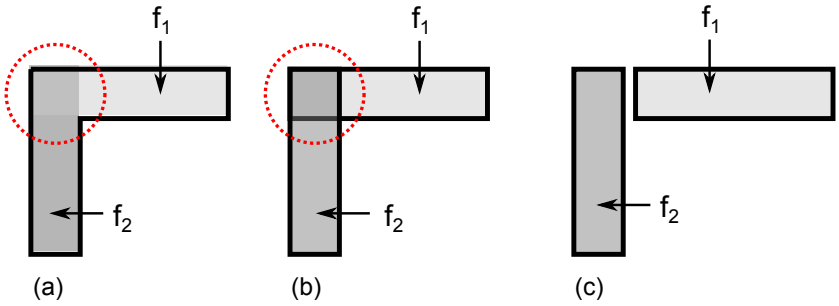


Figure 6.9: Separation of Feature Tool Bodies

Figure 6.9a shows a model with two features f_1 and f_2 with "Union" type of boolean between them. Once the boolean type is changed, the tool-bodies of features get isolated, even though they are placed at the same location. Figure 6.9b shows that there is no connection left between them anymore. Although Figure 6.9c depicts feature tool bodies in exploded view.

Thus, at this stage, all feature tool bodies are separated, however, spatially they may still overlap each other.

- **Concave edge partitioning (CEP)**: Even after separating the feature tool bodies, there could be volumetric overlaps as seen in Figure 6.9b. In such cases, at this step, overlapping feature volumes are split at the concave edges, by a technique known as Concave edge partitioning (CEP) or Convex Partitioning (CP). It is a well established technique [22, 55, 92, 117, 118], where faces incident at a concave edge, are extended and used as cutting geometries to split the model.

  Figure 6.10a shows the the original input model. Decomposition is done at the concave edge by extending faces incident at it, as cutting geometries. Figure

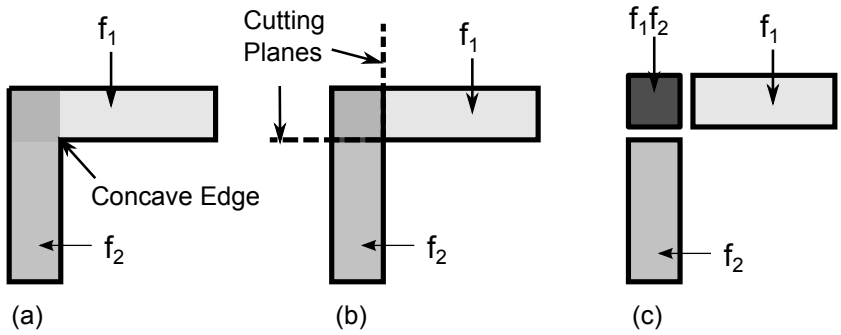
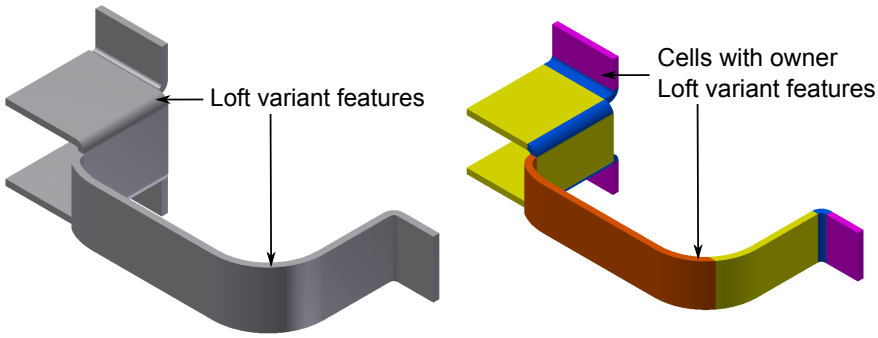


Figure 6.10: Concave Edge Partitioning

6.10b shows the cutting geometries (cutting planes in 3D). If the "Union" has been changed to ''New Body" already, as mentioned in the previous step, large extensions need not be done. The existing faces of the separated tool bodies themselves act as cutting geometries. Figure 6.10c shows decomposed cells in an exploded view. In the proposed work, the face extensions are not done infinitely (or going beyond the model's bounding box) but are restricted within the influence zone decided by the two interacting features. This enhancement avoids generation of a large number of redundant cells.

In some cases, the "L" shaped model as seen in Figure 6.10a, may get created, not by booleaning two features f_1 and f_2 but as a single feature, say f_3, with "L" shaped profile. In such cases, CEP is not applied to the concave edge in f_3. As shown in the next stages, f_3 computes its own midsurface patch by computing midcurve of the profile and then extruding it.

Out of the overall FBCD approach, the actual decomposition step (convex partitioning or CEP) has been leveraged from existing approaches, whereas the rest,i.e. feature assignment, localized partitioning, etc. are contributions of the present research work. Figure 6.11a shows an example of CAD model of a sheet metal bracket and Figure 6.11b is the output of FBCD, with each cell-bodies shown in different shadings.



(a) Input Model (b) Feature Based Partitioning

Figure 6.11: Feature-based Cellular Decomposition

Salient features of the proposed FBCD compared to the traditional cellular decom-

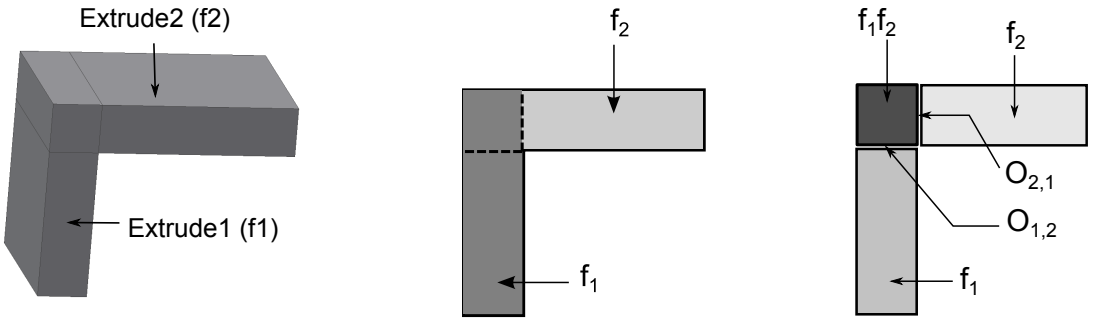
position approaches are:

- Traditional cellular decomposition works on the B-rep solid model with techniques such as Convex partitioning (CEP), whereas FBCD, as proposed here, works on the feature based CAD model.
- Cells in FBCD are assigned with the owner feature, but not so in the traditional cellular decomposition. Owner feature information is useful to certain algorithms like Hexahedral meshing [124] and also to the present research work for computing midsurface patches.
- In FBCD, the partitioning is not by simply extending all faces beyond the model and using them as cutting tools, rather it resorts to localized partitioning between two interacting features. This makes partitioning effect only in the local zone of interaction, thereby avoiding generation of large number of redundant cells.

The outcome of FBCD is the feature-based cellular topology which is used in the present research to compute the midsurface.

6.3.3 Formation of Feature-based Cellular Topology

FBCT represents a feature based CAD model as a connected set of cells with owner features. Two cells do not overlap volumetrically. Any two adjacent cells are separated by fully overlapping cell-faces.



(a) Original Model (b) Schematic Model (c) FBCT

Figure 6.12: Feature-based Cellular Topology

Figure 6.12a shows a representative case, schematically as Figure 6.12b, where two features f_1 and f_2 are interacting. Figure 6.12c shows that FBCD decomposes model having features f_1 and f_2 in such a way that a common cell, with owners $f_1''f_2''$, is formed. 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 ids of instances.

In the present research work as the FBCT is generated by decomposing $\mathcal{ABLE}$ features, the owner features of the cells are the $\mathcal{ABLE}$ features, i.e. Loft equivalents. Thus FBCD essentially decomposes the Loft feature tool bodies.

During each decomposition of the original Loft feature's tool body, its corresponding Loft parameters get split. Loft has two parameters, a $profile$ and a $guide-curve$. Decomposition can happen across any or both of these. Remaining parameter ranges are stored in the Loft features for each respective cells.

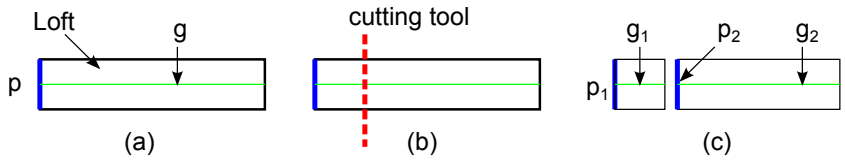


Figure 6.13: Loft Feature Partitioning

Figure 6.13a shows a single Loft feature getting decomposed and is represented by profile p and guide curve g. In this particular case cutting is happening on the guide curve g. Figure 6.13b shows the cutting geometry used to decompose. Figure 6.13c shows the outcome of FBCD, i.e. two cells, with profiles p_1 and p_2 along with the guide curves g cut into two g_1 and g_2, respectively. For both the cells, profile p is same as before but the parameter ranges of g are different, e.g the first cell has $p, g_{0,0.3}$, whereas the second cell has $p, g_{0.3,1}$. Similar cutting can happen across $profile$ as well, where the guide-curve (g), may remain uncut.

Following subsection elaborates process of populating a graph of the FBCT cells.

6.3.4 Formation of Feature-based Cellular Graph

FBCT has, say n cells, each assigned with a Loft owner-feature. A cell adjacency graph [CAG, $G(n,e)$] is formed with n nodes, each pointing to and representing a cell. Figure 6.14 shows the CAG of simple cells-configuration of a "L" shaped part. Node n_1 corresponds to a cell, with owning feature f_1', whereas node n_3 corresponds to the cell with owning feature f_2'. The common cell owned by $f_1'' f_2''$ is represented by node n_2. Note that the terms "node" and "cell" are used synonymously hereafter, unless specified otherwise. Each boundary face between two nodes is represented by an edge (e). Edge e_{12} corresponds to the overlapping face O_1, where as edge e_{23} corresponds to the overlapping face O_2.

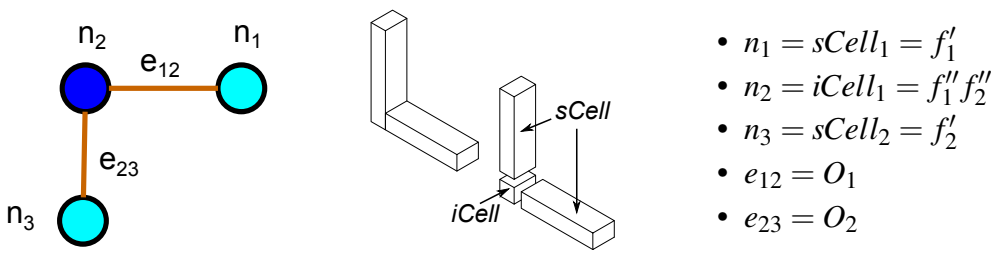


Figure 6.14: Feature-based Cellular Graph

Following section elaborates classification of the cells, which helps in delegating specific tasks to each in the proposed approach of computation of the midsurface.

6.3.5 Classification of Cells

Based on the expectations from midsurface shown in Figure 2.2 in Chapter 2, it can be seen that, the model can be conceptually divided into two portions with different functionalities, viz midsurface patches generating portions and midsurface patch joining portions. When this observation is mapped to CAG shown in Figure 6.14, it is clear that nodes n_1, n_3 are patch generating nodes and n_2 is a patch joining node. Thus, nodes are classified as *patch* nodes and *junction* nodes respectively. This classification is done based on the graph topology and characteristics of the owner feature parameters of the cells, these nodes point to.

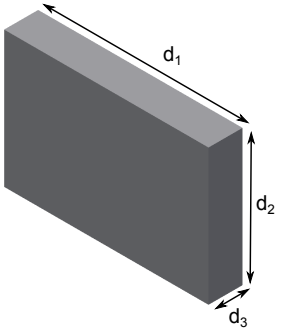


Figure 6.15: Cell with Dimensions

Cells are classified as solid cells (*sCell*s, pointed by *patch* nodes) and interface cells (*iCell*s, pointed by *junction* nodes). Figure 6.15 shows a cell with dimensions. Its classification rules are as follows:

Definition 3 *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* $d_1 < t \times d_2 \quad \& \quad d_1 < t \times d_3$.

Definition 4 *Interface cell (iCell) is a cell-body with minimum two dimensions (d_1, d_2) less than the threshold-factor (t) times of the maximum dimensions (d_3), denoted as* $d_1 < t \times d_3 \quad \& \quad d_2 < t \times d_3$.

In Figure 6.14, nodes n_1 and n_3 point to cells which are solid cells (*sCell*), whereas node n_2 points to cell which is interface cell (*iCell*).

The fundamental rule of midsurface computation is stated by Def. 5 as:

Definition 5 *sCell computes midsurface patch, whereas iCell connects all the midsurface patches incident on it.*

Advantage of the CAG representation is that, it is easier to classify nodes based on cell characteristics and delegate specific computational work to each using generic rules. Another advantage is that, as the problem space has been decomposed into manageable sub-problems.

Following sections elaborate the work delegated as per Def. 5 to the classified cells.

6.4 Generating Midsurface Patches from Solid Cells

This section presents an algorithm for computing a midsurface patch from a solid cell (*sCell*). The patch is generated based on the profile p and the guide curve g of the owner loft feature of the *sCell*. Depending on the aspect ratio of the *sCell*, midsurface patch computation varies. Two method are available, namely Midcurve based and Offset based. Midcurve based method is suitable wither the profile is "thin" and has a long guide curve, whereas the offset method is suitable where guide is shorter compared to the profile. Thus the input *sCell* is first classified and the accordingly midsurface patch computation method is chosen. Thus, the *sCell* is classified into long guide, short guide and equal guide.

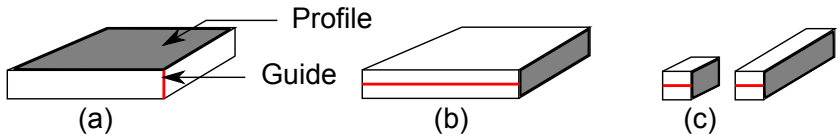


Figure 6.16: Types of *sCell*s

Figure 6.16a shows an Extrude feature (a Loft equivalent) having a profile and a guide curve where guide is far shorter than the size of the profile. This case is called as "Short Guide". Profile, being a 2D entity, its comparison with 1D guide curve can not be done directly. So, 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*. As mentioned in Section 2.1, though the criterion for "thinness" depends on the domain, $thinness = 2$ is considered for the present research. Figure 6.16b shows a similar Extrude but which is built by extruding smaller profile along longer guide direction. This case is called as "Long guide". Figure 6.16c shows case of Extrudes whose profile and guide sizes are similar. These cases are called as "Equal Guide" and the cells as "thick cells".

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

- **Long Guide** *sCell*: *Midcurve* is extracted from the *profile* and swept along the *guide* to generate the midsurface patch.

  Figure 6.17 shows a solid cell having a profile, a guide, a midcurve computed

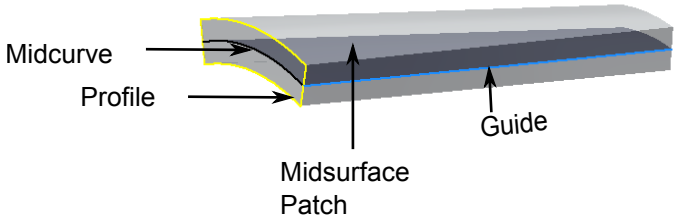


Figure 6.17: Long Guide *sCell*

and the midsurface patch generated. Details of computation of the midcurve are presented in the next section.

- **Short Guide** *sCell* : The *profile* face itself is *offseted* along half *guide* length. *Midcurve* is not computed in this case.
- **Equal Guide** *sCell*: This is a thick cell and thus midsurface patch is not generated for it. During midsurface patch joining phase, relevant thick cells may participate as patch joining cells.

Algorithm 8 describes the midsurface patch generation process in a pseudo-code.

Algorithm 8 sCell Midsurface Patch Computation

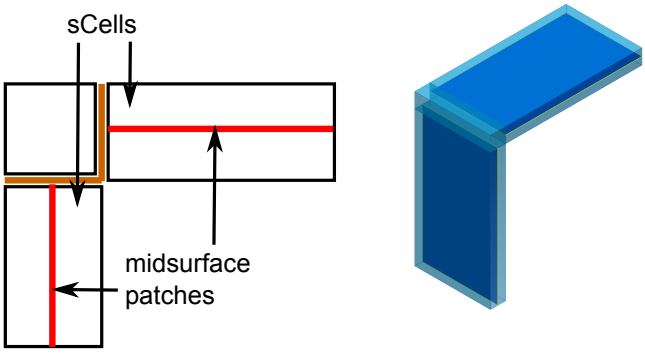
Require: *sCell*

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: $\quad m^1 = p \rightarrow compute\_midcurve()$
6: $\quad m^2 = sweep(m^1, g)$
7: **else if** $is\_short\_guide(p,g) == true$ **then**
8: $\quad m^2 = offset(p, g/2)$
9: **end if**

Steps for the same are elaborated below:

1. Owner feature is queried from the input *sCell*. From the owner feature, which is a Loft-equivalent, its profile and guide are extracted (Algorithm 8 lines: 1-3).
2. Based on the relative size of profile with respect to guide, the *sCell* is checked if it is Long-guide or a Short-guide cell.
3. If it is a Long-guide case, then midsurface patch is computed by "midcurve' method.
4. Midcurve from the profile is computed and it is swept along the guide to create a midsurface patch (Algorithm 8 lines: 5-6).
5. If the cell is of Short-guide type then "offset" method is used.
6. The profile face is offseted at half the distance of the guide to compute the midsurface patch (Algorithm 8 line: 8).

Figure 6.18 shows the outcome of this midsurface patch generation module with a sample "L" shaped model. FBCD decomposed it into 3 cells, out of which 2 are *sCell*s. The first picture shows schematic diagram of the two *sCell*s having midsurface patches generated in each of them. The cell in between is an *iCell*, so it is not handled in this module. The second picture shows the same scenario as a 3D view, showing the two midsurface patches.



(a) Patches, 2D view (b) Patches, 3D view

Figure 6.18: Midsurface Patches

Once done, all the *sCell*s are filled with the midsurface patches and are ready to get connected in *iCell*s, which is eleborated in Section 6.5.

Following section describes the proposed approach of generating midcurve from Loft feature's profile, needed in case of "Long Guide" *sCell* case.

6.4.1 Generation of Midcurves from Profile

This section presents a proposed approach of computing midcurve of a given profile, in case of Long-Guide *sCell*. The owner feature of this cell, the Loft, has longer guide curve compared to size of its profile. It needs to be noted that in feature based CAD paradigm, sketch consists of one or more profiles. There is one outer profile which represents the boundary of the sketch and the remaining profiles represent inner holes in the sketch. When there is only one i.e. outer profile, the terms sketch and the profile are used synonymously.

Midcurve is a curve, which lies midway of a profile. It, being lower in dimension than the input shape, applications like pattern recognition, approximation, similarity estimation, collision detection, animation, matching and deformation can be performed efficiently on it compared to them done on the original input shape.

Figure 6.19 shows two examples. Figure 6.19a shows a "Y" profile, with its midcurve whereas Figure 6.19b shows a "L" profile with its midcurve. Note that, both midcurves are one dimension less and lie midway of the respective profiles.

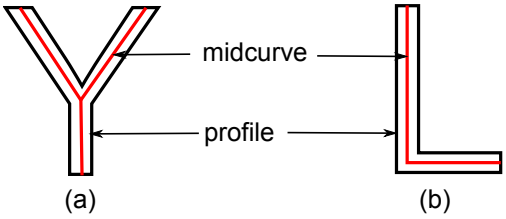


Figure 6.19: Examples of Midcurves

Most of the midsurface generation approaches reviewed in Chapter 2 in Section 2.4.5 are applicable to midcurve generation as well. Only difference is, instead of a 3D solid model as input for midsurface generation, a 2D profile is the input for midcurve generation. Approaches such as Medial Axis Transform (MAT), Chordal Axis Transform (CAT), Thinning etc. are used to compute, a generic curve form, known as medial curves. Midcurve is one specialization of the medial curve. Figure 2.11b had shown midcurve output where spurious branches have been replaced by two extensions forming a continuous line, thus mimicking the original shape.

Ramanathan [8] states that no formal definition exists for either midcurve or midsurface. He attempted to define midsurface semi-formally as seen in Definition 1. Similarly midcurve can also be defined as:

Definition 6 *Midcurve is an aggregation of curve segments (where each segment corresponds to a pair of nonadjacent edges in the object that are closest to each other) that form a closed and connected set and that satisfy homotopy*

Review of the reported approaches of computing medial curves is presented in Chapter 2 in Section 2.4.5. It suggests to avoid formal methods such as MAT, CAD, Thinning and Parametric, for computing midcurve as they need heavy post-processing to remove unwanted curves. The heuristic method of decomposition, which has been error prone and inefficient so far, appears promising in case of profile decomposition if enhancements can be proposed.

Following section takes a closer look at some of the existing profile decomposition approaches. After that, existing midcurve generation approaches based on profile decomposition are also reviewed.

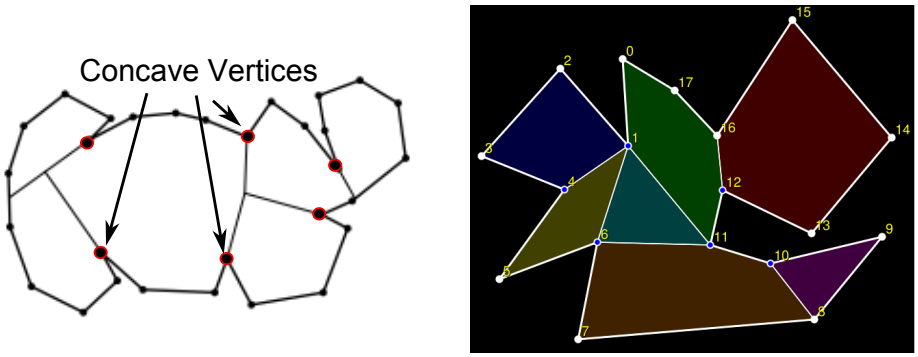
6.4.2 Related Work

Many approaches assume the input profile to be of simple polygon type, meaning, all the profile curves are linear and non-intersecting. The polygonal profile is assumed to be a set of connected lines end to end and closing the loop.

Keil [125] presented an approach based on convex partitioning, i.e. partitioning polygon at concave vertices, with an intent of making sub-polygons, convex. Figure 6.20a shows how his approach finds all possible ways to remove concavity of vertices and then takes the one that requires fewest diagonals.

Lien et. al. [26] decomposed polygons 'approximately' based on iterative removal of the most significant non-convex feature.

Bayazit [126] presented a polygon decomposition method based on Concave (Reflex) angle partitioning. Figure 6.20b shows the partitioning at specific concave vertices. Although the output was far better than if usual meshing had been employed, the drawback was it left out some corner cases giving more than necessary divisions.



(a) Keil [125] (b) Bayazit [126]

Figure 6.20: Polygon Decomposition Methods

Rocha [127] [128] presented skeletonization approach for images which primarily worked on vertices. Figure 6.21 shows how the sub-shapes computed mid-segments which were joined at the connections. Although this approach could address many shapes, it lacked comprehensive coverage and did not do very well for simple joints like T and L

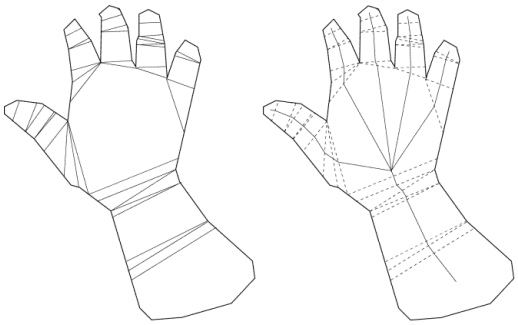


Figure 6.21: Midcurve Computation after Polygon Decomposition

Review of the approaches suggests that the polygon decomposition rules are needed to take into account the application context, accuracy, characteristics and aspect ratio of the sub-polygons. For the present research work, the midcurve generation needs primitive shaped, thin sub-polygons. Non-primitive, skewed shapes would result in inappropriate midcurves.

Following section proposes midcurve computation approach based on profile decomposition.

6.4.3 Proposed Approach for Generating Midcurve

In previous chapters it is seen that the proposed midsurface computation approach, first simplifies the input model by defeaturing, unifies the feature representation by generalization and then decomposes the simplified model into cells. Cellular topology makes devising generic approach for computing midsurface, easier and deterministic. Similar approach can also be used in computing midcurve from a profile.

Following is the proposed approach for computing midcurve which takes the input profile though similar stages as that of computation of midsurface.

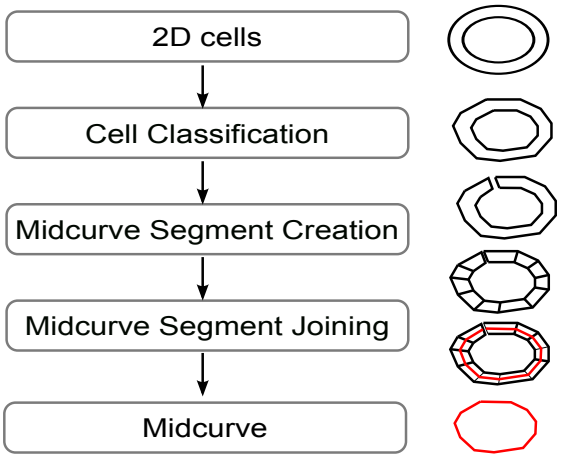


Figure 6.22: Overall Wokflow of Proposed Midcurve Computation

Figure 6.22 shows the stages which are explained below:

1. Input to this approach is a profile, of a generalized Loft of $\mathcal{ABLE}$ paradigm, who is an owner feature of a *sCell* for which a midsurface patch is being computed. The profile is typically made up of a variety of curves, such as lines, arcs, splines, etc. joined end-to-end forming closed loops.
2. During simplification of the profile, the curves are faceted into linear segments. Thus the input profile becomes a polygon. This is termed as “Polygonization”.
3. Polygon decomposition approach needs a single polygon whereas Loft feature’s sketch may have inner profiles representing holes. These inner profiles are connected to outer boundary profile via bridge curves. Thus multiple profiles get converted to single continuous polygon. This is termed as “Unification”.
4. During decomposition the polygon is split into sub-polygons, called 2D cells.
5. The set of connected 2D cells is used to compute the midcurve.

Amongst the stages shown, Polygonization and Unification are already established techniques. Decomposition and Midcurve Computation are the two areas in which the present research work has contributed. Following subsections elaborate them in details. Input to them is a polygon which is arrived at after the “Polygonization” and “Unification” processes, as mentioned above.

6.4.4 Polygon Decomposition

Polygon decomposition is the partitioning of polygons into primitive sub-polygons. Strategy for 'where-to-partition' depends on the application's need. Decompositions are done in such a way that minimum number of convex sub-polygons produced or total length of the boundary is minimized, etc. Convex Partitioning is one of the most popular method of polygon decomposition where partitioning happens at vertices having concave angle. Thus, as the concavity at those vertices is removed, after partitioning, the sub-polygons generated are of convex type having all vertices with convex angle.

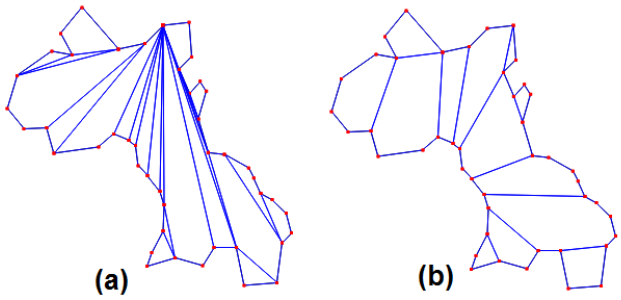


Figure 6.23: Convex Partitioning (Source: CGAL [25])

Figure 6.23a shows one of the possible polygon decomposition approaches whereas Figure 6.23b shows Convex Partitioning. The present research proposes to go in the direction of Convex partitioning but with an enhanced approach as elaborated below.

The objective of polygon decomposition by convex partitioning method, is to remove ''concavity" of the polygon. Figure 6.24 shows examples of concave and convex polygons with a simple line test.

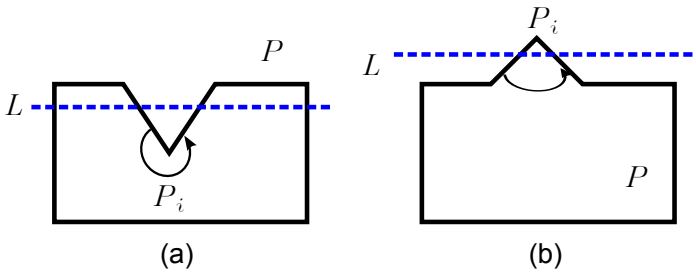


Figure 6.24: Concave and Convex Polygon

A polygon with any of the internal angles greater than 180 degrees is known as a concave polygon. As seen in Figure 6.24a, if a line (L) passing through the polygon (P) cuts more than two places then its a concave polygon. This is due to > 180 angle at vertex P_i. Such vertices are known as Reflex or Concave vertices and their presence is termed as "concavity" of the polygon. Figure 6.24b shows a case where L cuts P only at two places. It is a convex polygon. It is so due to all vertices having angles < 180. All such vertices are termed as convex vertices.

The objective of Convex Partitioning is to remove "concavity" of the polygon. Fig-

ure 6.25a shows the overall process. Concavity of the polygon P is computed by detecting reflex vertices. Polygon is decomposed such that at-least one reflex vertex (r) is turned into convex vertex. If such vertex is found, P gets decomposed into two sub-polygons P_1, P_2. Each of them goes through the same process as original P, till concavity of all the (sub)polygons is removed. Figure 6.25b shows how the recursive decomposing takes place.

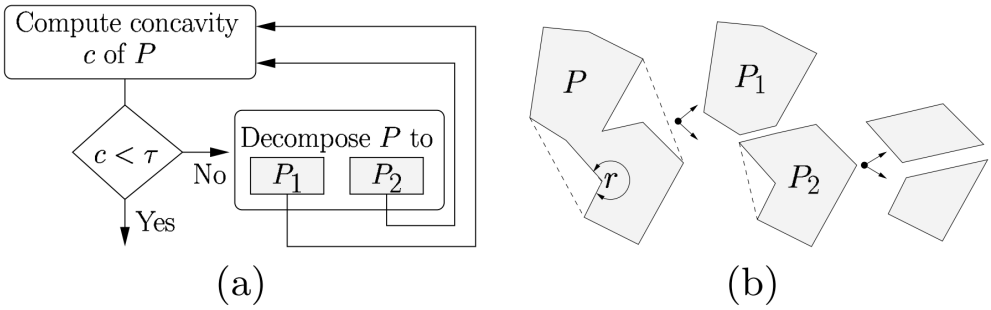


Figure 6.25: Convex Partitioning by Recursion (Source: Lien [26])

Polygon decomposition algorithm proposed in the present research work is based on existing convex partitioning method presented by Bayazit [126].

Following are the steps of the existing approach. Enhancements done to it are explained in the end. The comparison between Bayazit's approach and the present research approach is also presented.

Steps to decompose a polygon are:

1. Input is a simple polygon, denoted by P. It is defined as a list of vertices P_i (where, $i = 1 \rightarrow n$) ordered in a counter-clockwise manner. So, $P = \{P_0, P_1, ..., P_{n-1}\}$

   Polygon P can also be defined in terms of a set of connected edges, as:

   $P = \{\overline{P_0P_1}, \overline{P_1P_2}..., \overline{P_{n-1}P_0}\}$

   A polygon is simple if none of the edges intersect other edges anywhere-else other than at the shared endpoints of adjacent edges. This condition is denoted as:

$$\forall \quad \overline{P_iP_j}, \overline{P_kP_l} \in P, \begin{cases} \overline{P_iP_j} \cap \overline{P_kP_l} = \phi, j \neq k \\ \overline{P_iP_j} \cap \overline{P_kP_l} = P_k, j = k \end{cases}$$

2. All the vertices of the polygon are iterated one by one in counter-clockwise manner.

   Figure 6.26 shows simple polygon with 7 vertices, ordered counterclockwise. The current vertex is set as P_i and the direction of traversal is shown by the arrow.

3. P_i is checked if it is a reflex vertex by measuring angle r.

   Figure 6.27 shows that P_3 is reflex, as per definition below:

   Let $P_{j-1}, P_j, P_{j+1} \in P$, if the interior $\angle P_{j-1}, P_j, P_{j+1}$ is greater than π then P_j is a

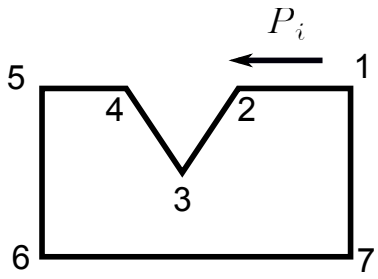


Figure 6.26: Polygon Traversal

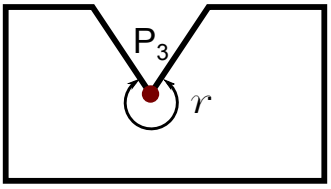


Figure 6.27: Polygon Reflex Vertex Detection

concave or *reflex* vertex.

Another way to test reflexivity or concavity of a vertex is if $Area < 0$, where $Area$ is a signed quantity and is as defined below:

$Area$ formed by three vertices in order (P_{j-1}, P_j, P_{j+1}) or two consecutive *edges* $(\overline{P_{j-1}P_j} \cap \overline{P_jP_{j+1}})$ is a signed quantity which is given by:

$$\begin{aligned} Area = {} & P_{j-1}.X(P_j.Y - P_{j+1}.Y) + \\ & P_j.X(P_{j+1}.Y - P_{j-1}.Y) + \\ & P_{j+1}.X(P_{j-1}.Y - P_j.Y) \end{aligned}$$

P_3 being a reflex vertex is termed as R.

4. Objective is to decompose the polygon at R such that its concavity is removed and the two sub-polygons that get created will have convex angles at respective vertices which originally was R. So, for cutting the concave angle at R, lines incident at it are extended. The piece of polygon in between the intersection of extended lines is called as "Range".

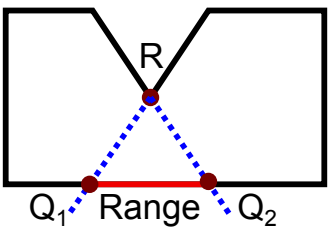


Figure 6.28: Range Detection

Figure 6.28 shows that R is the reflex vertex. Lines incident at it, i.e. the line coming into P_i and going out of P_i, are extended till they intersect remaining of the Polygon, say at Q_1 and Q_2. The segment within Q_1 and Q_2 is called *Range*.

5. Now the intent is to find a vertex so that line from R to it can be used for partitioning the polygon. Following are the cases depending on the number and types of vertices found in the *Range*.

   (a) **None**: If there are no vertices found in the range, or any of the polygon vertices themselves are end vertices of the range then a new vertex is created in the middle of *Range*. This new vertex is called as "Steiner" vertex.

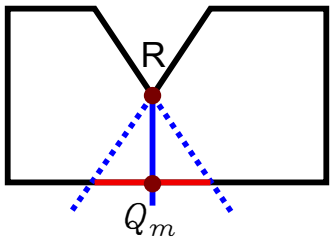


Figure 6.29: Steiner Vertex Creation

   Figure 6.29 shows that there are no vertices in the *Range* so a Steiner vertex Q_m is created and thus line $R - Q_m$ becomes the cutting or partitioning line, called "chord", used to split the polygon.

   (b) **Closest Reflex**: If there are multiple reflex vertices in the *Range* then the closest amongst them gets the highest priority to form the chord.

   (c) **Reflex**: If a vertex found in the *Range* is a sole reflex vertex then its gets the next priority to form the chord.

   (d) **Closest**: If non of the vertices found in the *Ranges* are reflex vertex, then closes amongst those gets the priority to form the chord.

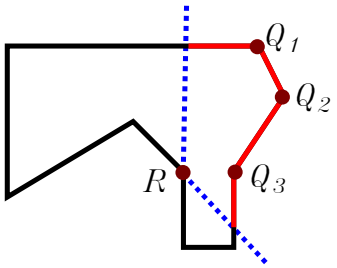


Figure 6.30: Vertex Priorities for Chord Creation

   Figure 6.30 shows a different profile shape examples having 3 candidate vertices in the *Range* to form the chord. Q_3 being the only reflex vertex, gets the chance to form the chord.

   (e) **Visible**: It is made sure that the selected vertex Q_3 is "visible" from the reflex vertex R. If it is not so, then this cut can not be made. The next best choice is chosen for evaluation. Visibility test is performed as defined below:

   P_k is visible from P_i if $\overline{P_iP_k}$ is a *diagonal* of P.

   where, "Diagonal" is defined as: $\overline{P_iP_k}$ is a *diagonal* of P. So, a *diagonal* is a line

segment between two vertices that only touches the interior of the polygon.

6. Polygon is divided at the chord. Figure 6.31 shows the formation of the chord between $R - Q_3$.

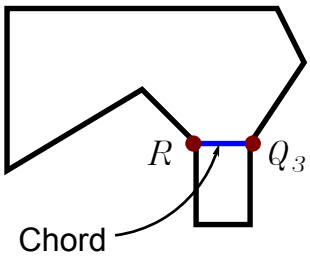


Figure 6.31: Polygon Decomposition by the Chord

7. The decomposed sub-polygons are sent through the same process recursively till there are no reflex vertices left.
8. Output of the above steps is a set of connected sub-polygons.

The proposed approach improves upon the Bayazit's algorithm [126] stated above, in terms of expanding search to include even the extreme vertices in the range, thereby giving minimal and elongated sub-polygon shapes. Midcurves are typically for thin, elongated shapes. Thus the proposed improvement results in the sub-polygons of shape characteristics needed for computation of midcurves.

Following is the list of salient improvements over Bayazit's algorithm [126]:

1. If there are any vertices at the ends of the *Range*, they were getting ignored, or were sent for "None" (i.e. Steiner vertex) case. The proposed approach includes them as candidates for further evaluation based on the priorities stated above.

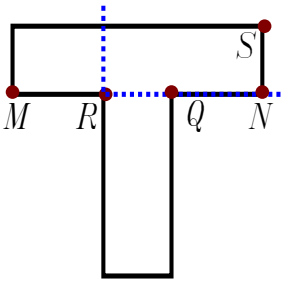


Figure 6.32: Including Extreme Range Vertices

Figure 6.32 shows that incoming edge (MR) is hitting the end vertices of the test-line (QN) or is collinear, it (Q) was getting ignored in the existing algorithm [126]. In that case the next closet vertex (S) was getting chosen. This was corrected in the proposed algorithm and a shorter cut with chord RQ is done.

2. The midcurve creation algorithm requires these sub-polygons to be of two primitive types viz. triangles and quadrilaterals. So, if any of the sub-polygons has more than 4 sides then those sub-polygons are are triangulated with Constrained Delaunay Triangulation (CDT).

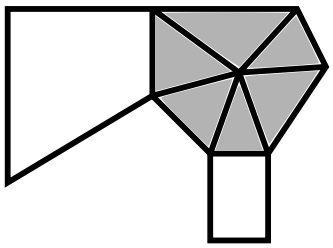


Figure 6.33: Triangulation of More Than 4 Sided Polygons

Figure 6.33 shows a sub-polygon with more than 4 sides has been triangulated.

Algorithm 9 shows the proposed approach of polygon decomposition.

Algorithm 9 Polygon Decomposition

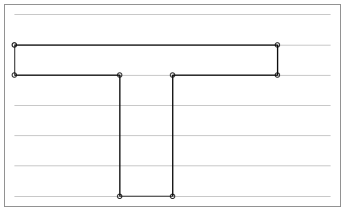
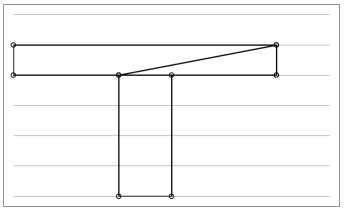
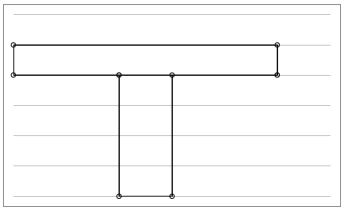
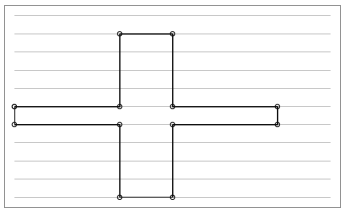
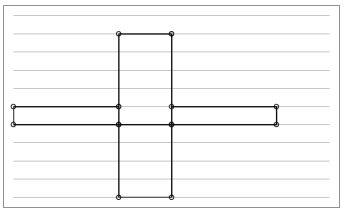
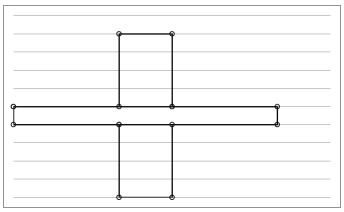
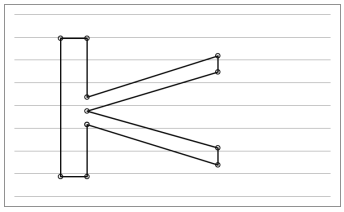
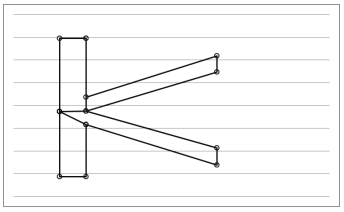
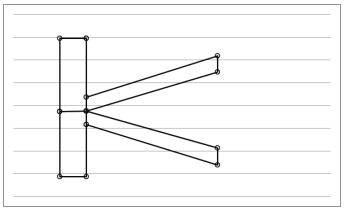
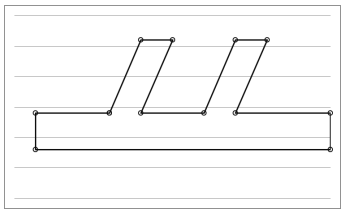
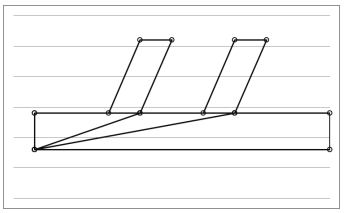
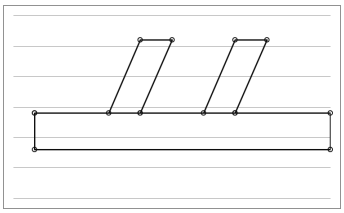
Require: 2D Planar polygon represented by list of vertices
Ensure: Vertices in counter-clockwise direction

```
1:  while End of vertices list has not reached do
2:      Get the current vertex.
3:      if current vertex is a Reflex vertex R then
4:          Extend the edges incident at R until they hit an edge
5:          if Extension line and Polygon side are collinear then
6:              Find closest vertex which is not internal to the extension line
7:          end if
8:          if there are no vertices to connect to then
9:              choose a vertex in the middle
10:         else
11:             Find vertex to connect to
12:             Find best vertex Q_i within the range, to form the partitioning chord
13:             Make sure Q_i is visible from R
14:         end if
15:     end if
16: end while
17: Split the polygon at the cutting chord (line RQ_i)
18: Repeat with sub-polygons till there are no reflex vertices left.
19: Triangulate polygons with more than 4 sides.
```

As the last step in the **Algorithm 9** triangulates the remaining polygons, this algorithm guarantees presence of sub-polygons with 3 and 4 sides only. Further algorithms, like the one mentioned below **Algorithm 10**, thus, needs to devise logic for only the triangles and quadrilaterals, making it less error prone and more deterministic. **Table 6.3** demonstrates improvements over Bayazit's [126] algorithm with examples.

Table 6.3 shows comparison of results of polygon decomposition between the existing Bayazit's [126] approach and the approach proposed in the present research work.

Table 6.3: Comparison of Proposed Polygon Decomposition Approach with the Existing One

Input Polygonal Profile	Bayazit's Approach	Current Research Approach
		
		
		
		

The output of the proposed approach has primitives of types triangles and quadrilaterals and no other redundant splittings. The resulting sub-polygons are sent for computing connected midcurves, as described in the following section.

6.4.5 Generation of Midcurves from Sub-Polygons

The objective of this module is to compute a connected midcurve from the set of sub-polygons received from the previous module. Following is the proposed approach of generating midcurve from 2D cells.

Figure 6.22 shows the stages which are explained below:

1. Set of 2D cells (sub-polygons) is the input. They are of only two types Triangles and Quadrilaterals.
2. Cells are classified into different classes based on type, i.e. triangle or quadrilateral and on the number of sides which are chords. A chord is a common interface-boundary shared between two cells. Each chord will have two sides owned by two different cells.

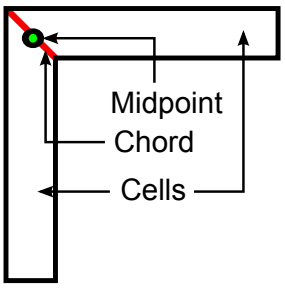


Figure 6.34: Chord Midpoint as Location for Joining Midcurve Segments

Figure 6.34 shows “L” shaped profile with 2 quadrilateral shaped cells, with a chord at the interface and its midpoint. Following steps make sure that midcurve patches from both sides of the chord are joined at the midpoint of the chord.

3. Midcurve segments are created based on the particular class of cells, elaborated in Tables 6.4, 6.5. ‘Thinness’ is an important criterion in choosing midcurves for the individual shape. Midcurves are generated along longer-length and not across shorter width.

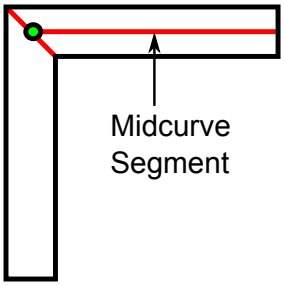


Figure 6.35: Midcurve Segment Creation

Figure 6.35 shows midcurve segment computed for one of the cells. In shapes like ’L’ midcurves from both sub-polygons, across the chord, join together at a vertex, naturally. Additional extensions are not required.

4. Midcurve patches are joined in cases where there is gap.

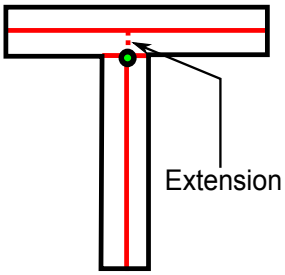


Figure 6.36: Midcurve Segment Extension

Figure 6.36 shows ’T’ shaped polygon. The horizontal midcurve does not connect with the common chord. In this case one of the midcurves is extended to join the other.

5. Output is a well-connected midcurve.

Following tables detail various cases and the process to compute midcurve segments.

Table 6.4: Triangle Cases of Midcurve Configuration

Shape	**Chords**	**Rule**	**Diagram**
Triangle	None	No Midcurve	
	One	Join Midpoint of the shorter side	
	One	Join Opposite vertex if both sides are of similar length	
	Two	Join bisectors	
	Three	Join to centroid	

Algorithm 10 presents the proposed approach for midcurve computation from 2D cells in a pseudo-code format.

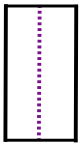
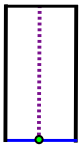
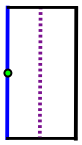
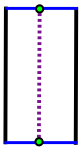
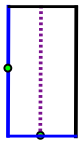
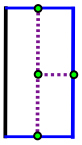
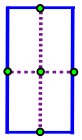
Algorithm 10 Midcurves Creation

Require: List of partitioned 2D Planar polygons represented by list of vertices in counter-clockwise direction

1: Find internal-common edges called chords
2: Iterate over all polygons and create chords at Full or Partial overlap
3: **while** End of Polygons list has not reached **do**
4: Get the current polygon P
5: Get chords which are part of P
6: Look at various configurations due to Num Sides and Num Chords
7: Generate Midcurves
8: Assign Midcurves on relevant side of the chord
9: **end while**
10: Extend chords which are not connected with other neighboring chords

Table 6.6 presents various cases demonstrating working of both, polygon decomposition as well as midcurve computation. It clearly demonstrates that the midcurve generated is well connected and represents the input profile shape faithfully.

Table 6.5: Quadrilateral Cases of Midcurve Configuration

Shape	Chords	Rule	Diagram
Quadrilateral	None	Find shortest side and create Midcurve in the direction average of both the adjacent sides	
	One (Shorter)	Create Midcurve in the direction average of both the adjacent sides	
	One (Longer)	Extend from Chord on longer side upto the midcurve	
	Two (Opposite)	Join midpoints	
	Two (Adjacent)	Ignore the chord on longer side and use one-chord rule	
	Three	Join to centroid	
	Four	Join to centroid	

6.4.6 Results and Discussions

Following are some of the examples taken from academic papers for benchmarking midcurve output against the proposed approach.

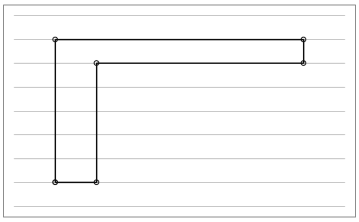
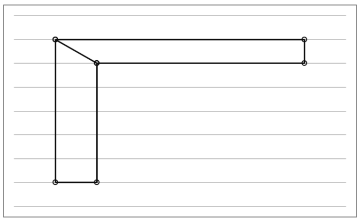
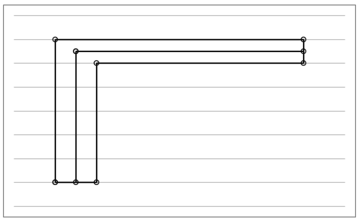
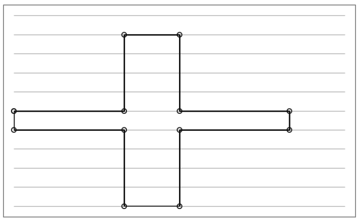
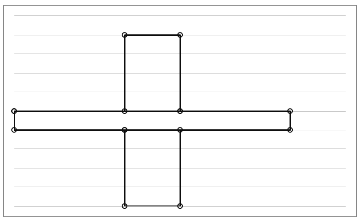
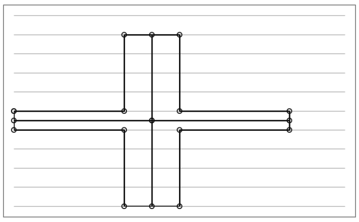
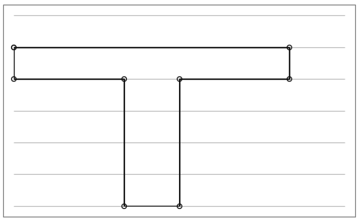
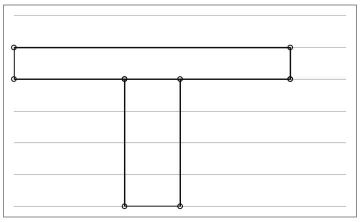
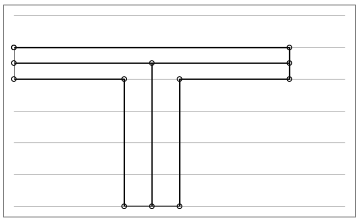
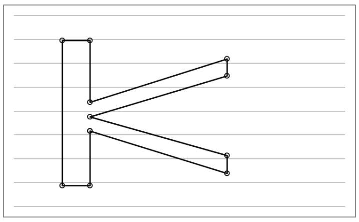
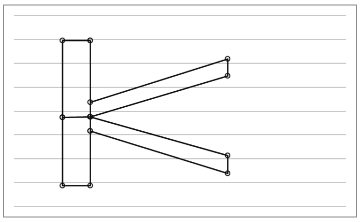
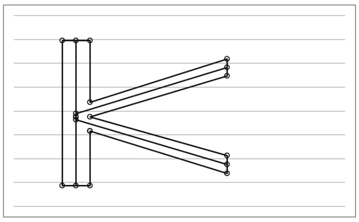
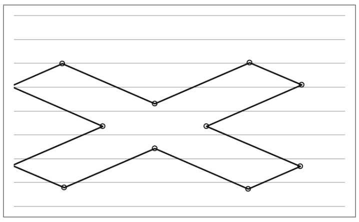
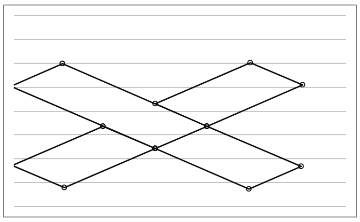
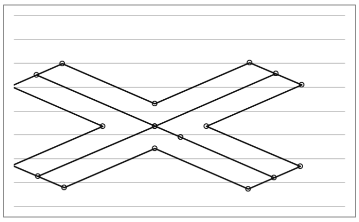
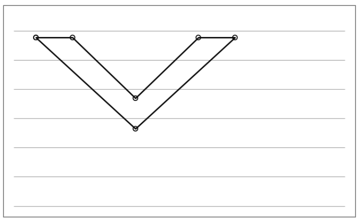
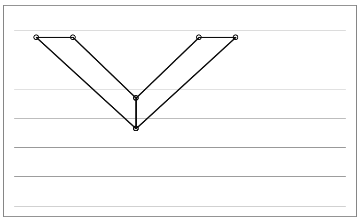
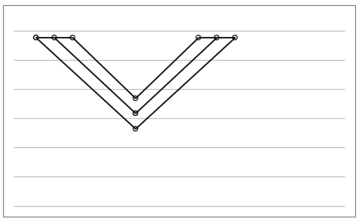
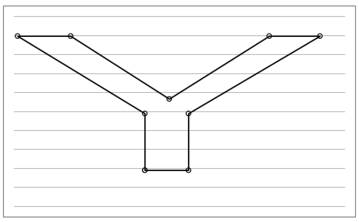
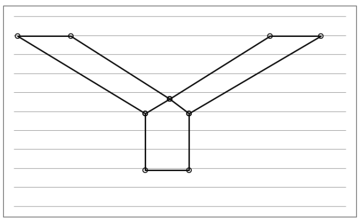
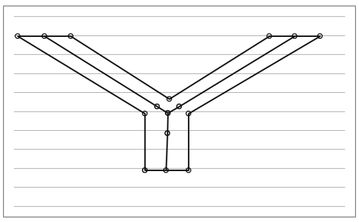
1. Glass profile was presented by Fischer et. al [85].

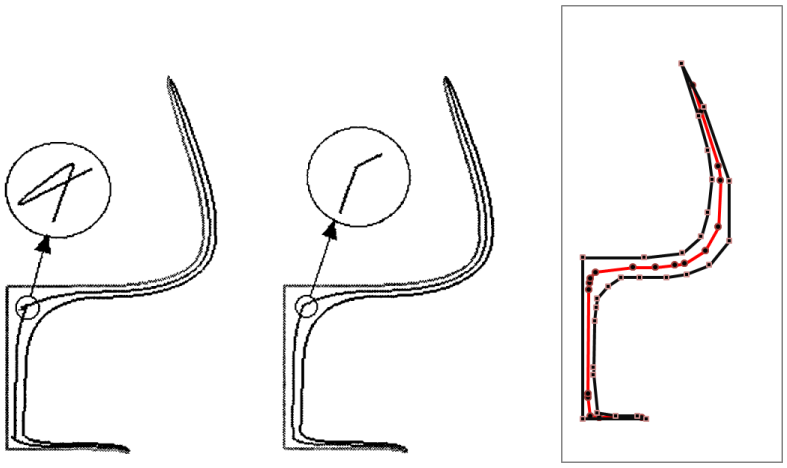
   Figure 6.37a shows an erroneous loop which had to be eliminated in the post-processing. No such post-processing is needed in the proposed approach and the output as seen in Figure 6.37b is appropriate. This example has been particularly chosen for showcasing midcurve of a profile, having free-form curves and variable thickness.

2. Pinto's [129] example of planar polygonal Horse profile.

   Figure 6.38a shows the midcurve (MAT) computed by Pinto's [129] approach whereas Figure 6.38b shows output by the proposed approach. It can be noted that all the un-

Table 6.6: Results of Partitioning and Midcurves Computation

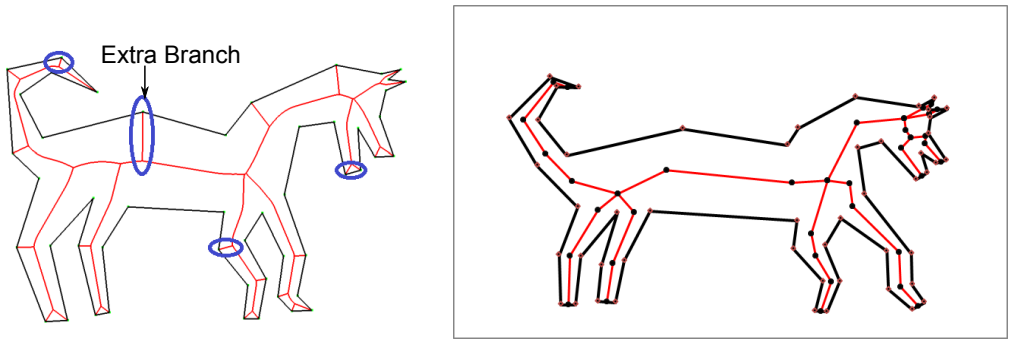
Shape	**Partitions**	**Midcurves**
		
		
		
		
		
		
		



(a) Loop Elimination (Source: Fischer [85])

(b) Proposed Approach

Figure 6.37: Midcurve of a Glass Profile



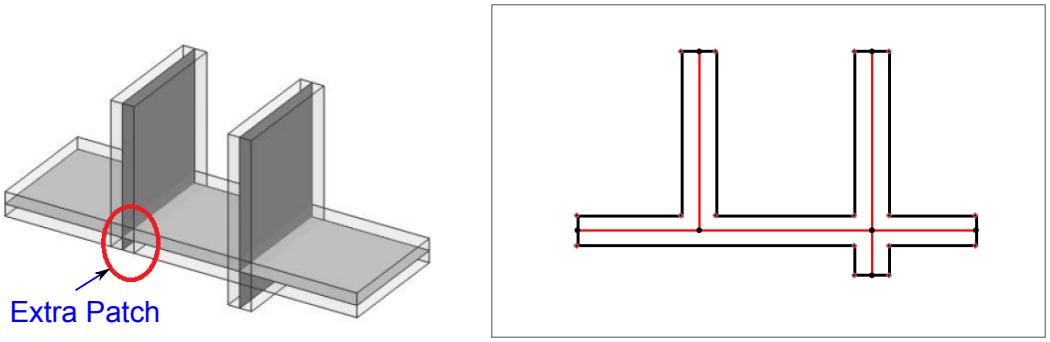
(a) Undesired Extra Branches (Source: Pinto [129])

(b) Proposed Approach

Figure 6.38: Midcurve of a Free Form Profile

desired branches at the corners have been eliminated.

3. Woo [3] presented following example to demonstrate appropriate extensions.



(a) Erroneous Patch (Source: Woo [3])

(b) Proposed Approach

Figure 6.39: Midcurve Extensions

Figure 6.39a is output of the traditional face pairing approach. Woo had to devise a logic to get rid off the invalid pairs creating such extensions. When mapped to

2D profile domain, it would result in an erroneous segment, as seen in the circle. Figure 6.39b shows the correct output by proposed method.

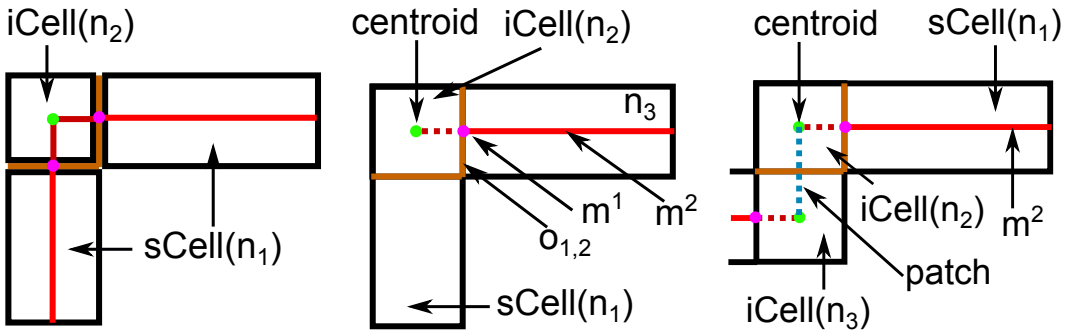
Midcurve computed by the proposed approach is used to compute midsurface patches for "Long Guide" *sCell* cases. After all the midsurface patches are computed for the *sCell*s, they are joined in the *iCell*s, as described in details in the following section.

6.5 Connecting Midsurface Patches in Interface Cells

This section continues after the midsurface patch generation approach mentioned in Section 6.4. So, at this stage, midsurface patches are ready at all the *sCell*s. Now this section presents the proposed approach for joining them in interface cells (*iCell*s).

*iCell*s have been delegated with the task to connect the midsurface patches incident on them, either by extending the midsurface-patches from the adjacent *sCell*s, or by generating new patches, so that joining of midsurface patches happens at common edges.

Each *iCell* is connected with adjacent cells via edges. Each edge has two nodes connected at ends. The one in which connections are going to be made is called the current node, the *iCell* it points to is termed as "self". The other node of the edge is called as "adjacent node". "Self" is always an *iCell*" but the"adjacent node" can be either a *sCell* or an *iCell*.



(a) Expected adjustments (b) *sCell* – *iCell* Interaction (c) *iCell* – *iCell* Interaction

Figure 6.40: Resolving of Overlap in *iCell*

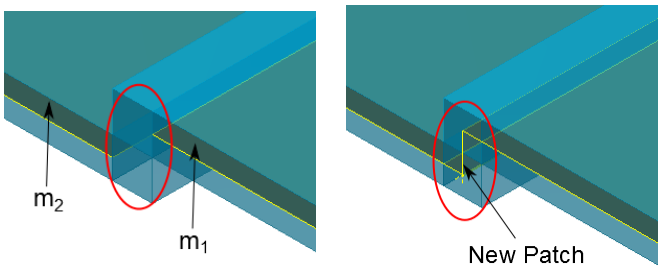
If the "adjacent node" is a *sCell*, then the interaction is of *iCell* – *sCell* type and if the "adjacent node" is an *iCell*, then the interaction is of *iCell* – *iCell* type.

Figure 6.40 shows working of both types of interactions viz. *iCell* – *sCell* and *iCell* – *iCell*. Figure 6.40a shows the input state of the model. It has two *sCell*s with ready midsurface patches. The expectation is that the patches will extend inside the *iCell* to join at the common edge. 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*. The extensions are shown by dotted line and the common

edge is shown as the point of intersection of the extensions.

Figure 6.40b shows the working of $iCell - sCell$ 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 6.40c 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 *iCell*s pairing with the other *sCell*. Centroids of both the *iCell*s need to be connected with a new patch.



(a) Before Resolving (b) After Resolving

Figure 6.41: *iCell* Resolving in Overlap Case

Figure 6.41 shows the $iCell - iCell$ interaction in 3D. Figure 6.41a shows the state before joining of the patches $m1, m2$ whereas Figure 6.41b shows state of the model with connected midsurface patches.Thus, all the *iCell*s join the midsurface patches to form a well-connected midsurface.

Algorithm 11 *iCell* Midsurface Patch Interaction Resolution

Require: *iCell*

while $size(iCell \rightarrow edges) > 0$ **do**
  $e_i = iCell \rightarrow edge$
  $n_s = iCell$
  $n_o = e_i \rightarrow get\_adjacent\_node(n_s)$
  if $n_o \rightarrow type == sCell$ **then**
    $m_o = n_o \rightarrow query\_midsurface()$
    $O_f = e_i \rightarrow overlapping\_face()$
    $m^1 = surf\_surf\_intersection(O_f, m_o)$
    $m^2 = extrude(m^1, n_o \rightarrow centroid)$
  else if $n_o \rightarrow type == iCell$ **then**
    $c_1^1 = get\_curve\_at\_centroid(n_s)$
    $c_2^1 = get\_curve\_at\_centroid(n_o)$
    $m^2 = create\_patch(c_1^1, c_2^1)$
  end if
end while

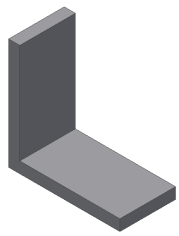
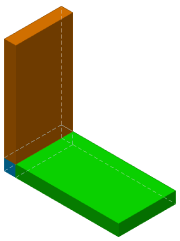
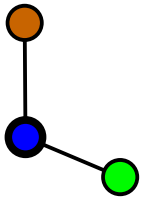
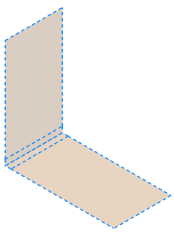
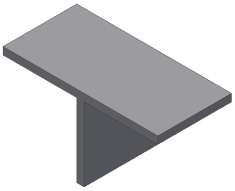
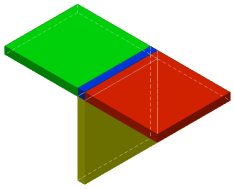
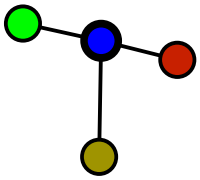
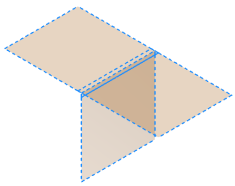
Algorithm 11 describes the midsurface patch joining process in a pseudo-code form. Steps for the same are elaborated below:

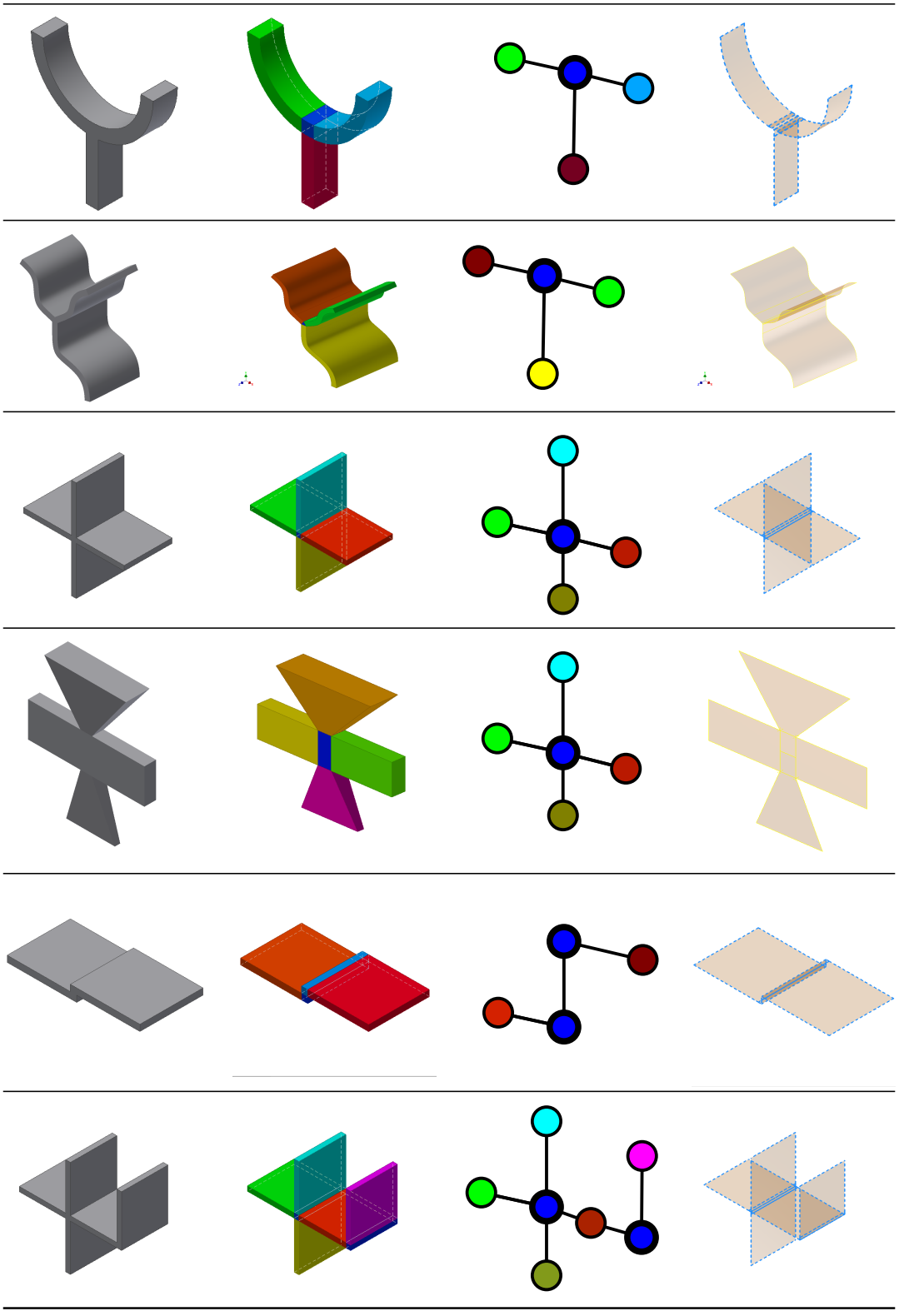
1. Input *iCell* is typically joined by one or more *sCell*s, via edges. Such incident edges are iterated one by one (Algorithm 11 lines: loop between 1 to 15).
2. Graph node at the other end of the edge is queried. Based on the type of the cell it points to further process is executed (Algorithm 11 lines: 2-4).
3. If the cell pointed by the other node is of type *sCell* then this interaction becomes $iCell - sCell$ type and further steps are stated below (Algorithm 11 lines: 5-9).
4. Midsurface patch from the other *sCell* is queried. Here one option is extend this patch up-to centroid of the input *iCell* and another option is to create a patch from the midsurface up-to centroid of the input *iCell*. As extension API needed for extension are not available, the later option is chosen in the present research work (Algorithm 11 line: 6).
5. Intersection between the midsurface patch and the face between $iCell - sCell$ is computed and then extruded up-to the centroid of the input *iCell* (Algorithm 11 lines: 7-9).
6. If the cell pointed by the other node is of *iCell* then this interaction becomes $iCell - iCell$ type and further steps are stated below (Algorithm 11 lines: 11-13).
7. A new patch is created between centroids of both the *iCell*s (Algorithm 11 lines: 11-13).

Once all interactions are resolved in *iCell*s, the result is a well-connected midsurface.

Table 6.7 shows various cases, with their respective CAD model, Cellular Decomposition (FBCD), Cellular adjacency graph (CAG) and midsurface outputs.

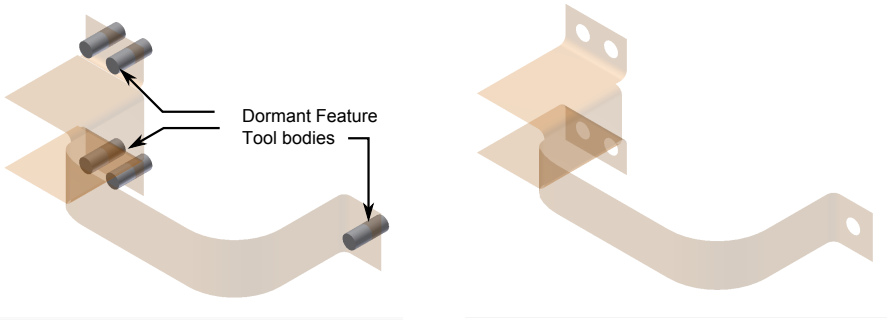
Table 6.7: Midsurfaces Generated from Feature-based Cellular Models

Model	FBCD	CAG	Midsurface
			
			



6.6 Re-application of Dormant Features

Midsurface generated is not yet final at this stage. As elaborated in Section 4.4.1, during defeaturing, negative features were temporarily removed even though they were relevant. The intent was to simplify and to make midsurface computation simpler. These removed features, called Dormant features, are reinstated back on the midsurface here. Cached dormant feature tool bodies are pierced into the midsurface.



(a) Dormant Features Tool-Bodies Piercing

(b) Final Output

Figure 6.42: Re-application of Dormant Features to Generated Midsurface

Figure 6.42a shows the tool-bodies being reapplied on the midsurface, whereas Figure 6.42b shows the output final midsurface.

6.7 Conclusions

This chapter presented proposed approach for computing midsurface from defeatured and generalized CAD model as input. It decomposed the model into cells having owner features. It populated a graph with the cells and classified them into midsurface patch generating cells and midsurface patch joining cells. Midsurface patches were generated either by offsetting profile face or by sweeping the midcurve of profile of the Loft feature, along its guide. Midcurve was computed by first decomposing the input profile, then computing midcurve for each sub-polygon and then connecting them. Midsurface patches were connected in midsurface patch joining cells to form a well-connected midsurface. At the end, dormant feature tool bodies were applied to re-apply the temporarily removed negative features.

It can be noted that the proposed approach, implemented in **MidAS**, has addressed the problems present in traditional approaches viz. computation of midsurface patches and joining them.

The next chapter presents proposed topological validation method for assessing the quality of output midsurface.

Chapter 7

Validation of Generated Midsurface

7.1 Introduction

This chapter presents an approach for validating the midsurface generated from a CAD model.

Midsurface, as defined by Rezayat [36], is expressed as a contiguous flow of the input solid (Definition 2). Midsurface is expected to lie midway and be representative of the input solid. The first part of the requirement, i.e. 'lying midway' is geometrical in nature whereas being 'representative' is topological in nature. Geometrical validation approaches check if the midsurface is at half the distance from side-faces of the input solid. Topological validation approaches check if the midsurface has similar connectivities between its sub-shapes as in the input solid. These validations help detect errors in midsufaces, such as missing midsurface patches, gaps or overlaps amongst them, midsurface not lying midway, etc.

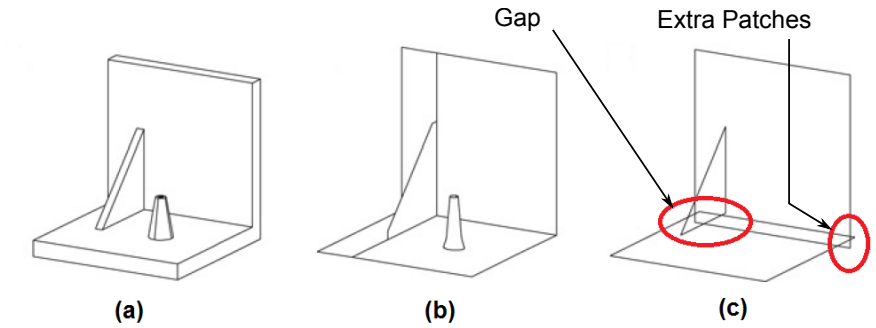


Figure 7.1: Model, Good Midsurface, Bad Midsurface (Source: Lockett [27]

Figure 7.1(a) shows an example input model. Figure 7.1(b) shows a correct output, i.e. a well connected midsurface. Figure 7.1(c) shows midsurface output with errors such as gaps and extra patches. Various approaches are used to detect such errors.

Traditionally, midsurface validations are done using following methods or tools:

- **Manual**: Manual inspection is done to ensure that the midsurface lies midway and is continuous throughout, especially at the junctions. If any errors are found,

such as gaps or overlaps, they are corrected manually by using available modeling operations such as extend or trim. This technique is obviously tedious, time-consuming and error-prone, because detecting errors and correcting them in complex parts is difficult.

- **Semi-automatic Tools**: Ready tools are provided by CAD-CAE systems which can detect gaps and overlaps, automatically. In some cases functionalities such as "auto-heal" or "auto-fill" are provided to put surface patches at the site of gaps or holes in the midsurface. These tools have limited applicability as the automatic detection and healing is not robust and does not work consistently for complex geometries. Also, these tools cannot detect correctness topologically, i.e. appropriateness of the connections of midsurface patches.
- **Automatic Distance Comparison**: Automatic tools are provided to check if distance between midsurfaces and the input solid is half the thickness. Hausdorff distance [27], which measures maximum dissimilarity between two shapes, is used for assessing geometric similarity. But, as pointed by Lockett [27], this technique does not work in some valid situations, such as at junctions, where the distances can be more than half the thickness.

All the above mentioned tools are used mainly for geometric validations. Following section evaluates some of the traditional validation approaches and establishes need for a new topological validation approach.

7.2 Need for Topological Validation

Review of the traditional techniques suggests that the 'Manual' and the 'Semi-automatic' techniques are cumbersome to use, especially in cases of large and complex midsurfaces. The automatic validation technique of distance comparison is thus a preferred method for complex midsurfaces. In this technique, Hausdorff distance between corresponding faces on the input solid are measured and checked if they are half of the given thickness.

Eqn 7.2 defines Hausdorff distance between two surfaces.

$$\bar{h}(A,B) = max_{a\in A} min_{b\in B} d(a,b)$$
$$H(A,B) = max(\bar{h}(A,B), \bar{h}(B,A))$$

$\bar{h}(A,B)$ gives maximum value amongst the least values of distances from sample points (a) on surface A to sample points (b) on surface B. The process is repeated in the other direction i.e. $\bar{h}(B,A)$. The maximum between these two values is the Hausdorff distance.

Figure 7.2 shows a sample computation of distance between a and b points on sur-

face A and B respectively. Minimum of these distances is computed first. Then maximum from all sample a points is considered as directed Hausdorff distance from A to B. Similarly directed Hausdorff distance from B to A is computed. Maximum between these two is considered as the Hausdorff distance. Thus it is a single representative value of distance between two surfaces.

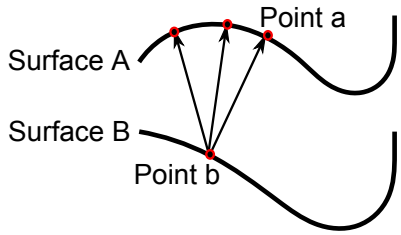


Figure 7.2: Hausdorff Distance between two surfaces

This technique of checking geometric similarity using Hausdorff distance is widely used in applications such as shape matching, validations, etc. But it has few drawbacks too. The accuracy of the method depends on the number of sample points. More the points better the accuracy but then, more are the distance computations. Apart from this, it does not particularly work at the junctions as the correct distance there can be more than the half of thickness.

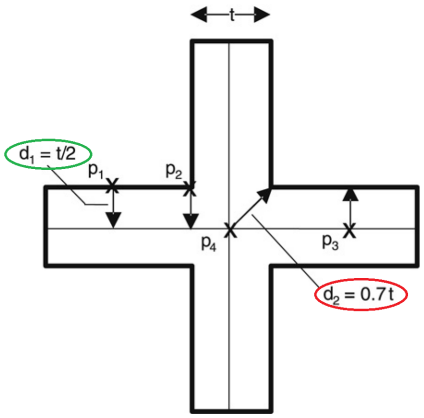


Figure 7.3: Distance at the junction (Source: Lockett [27])

Figure 7.3 shows that distance from points like p_1 and p_2 to midsurface is $d_1 = t/2$, where t is the thickness. But at the junctions, the distance from midsurface (point p_4) to its closest face on the input solid is $0.7t$. Although it is more than the permissible distance of $0.5t$, it is valid. Thus, it is not possible to ascertain the correct threshold for determining validation and so, for geometric validation techniques it is challenging to detect if the distance measured is appropriate or an error.

Geometric validation technique cannot find missing connections, similarity with the input solid, etc. Thus, to have more complete validation, geometric techniques need to be complemented by topological criteria as well. The present research proposes topological validation method in which topological entities of midsurface are predicted from

the input solid's topological entities. Validation is done by comparing the predicted entities with the actual entities of the midsurface. This technique helps find missing gaps, connections, etc. However, being topological, it does not carry out the essential geometric distance calculations, which the existing geometric validation technique can perform. The approach proposed here for topological validation complements geometric validation and thus makes overall validation of the midsurface more robust.

Following sections present the topological validation approach developed in this research work. It is however noted that this approach is limited to only providing a theoretical framework and is not iimplemented in the software system, **MidAS**.

To understand the state of the art, relevant works in the domain of topological validation are reviewed in the next section.

7.3 Related Work

In past, shape matching applications were being used to find similar parts from CAD databases. Criteria used for checking shape similarity can be classified as geometric and topological. Shape signatures, histograms are geometric, whereas graph based, feature recognition techniques are topological in nature [27].

In CAD, surfaces are represented by non-manifold topological equations. Lipson [130] derived equations related to topological invariant for the sheet metal non-manifold model which can be used for topological representation of the midsurface.

Lee [131] presented reverse of midsurface generation approach, called 'sheet thickening'. In this approach, a sheet (which can be considered as a midsurface) was offsetted, thickened to form thin-walled model. This work lacked detailed transformation equations.

One of the most significant contribution in the shape similarity assessment, in the context of midsurface, is by Helen Lockett [27]. She developed a methodology to predict topological entities of midsurface by using proximity groups adjusted by angle criterion.

Figure 7.4 shows Lockett's topological similarity validation technique. For example, in case of "L", at junction, the solid model has 2 edges, one concave (Cv) and one convex (Cx). Out of which, the number of Cv edges is 1. Angle at Cv is 90, so the predicted faces joining at the junction in the midsurface are 2.

In case of "T", as the sum of angles is 180, one more edge is added, making the number of edges at the junction as 3. But in case of, "Y", which is topologically similar to "T", as the sum of angles is 360, no additional edge is used, making the number of edges at the junction as 3. Thus use of hard-coded rules based on geometric angle values, makes this approach heuristic.

Example solid and mid-surface model edge attributes

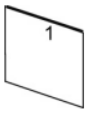
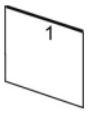
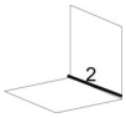
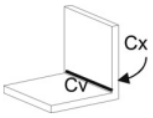
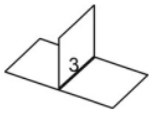
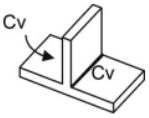
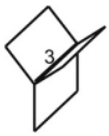
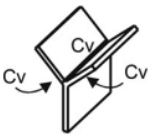
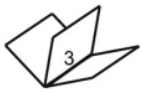
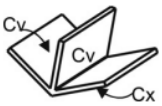
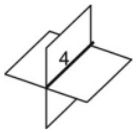
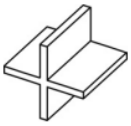
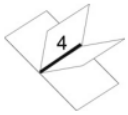
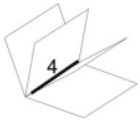
Connection type	Mid-surface model showing shared edge order (O_i)	Solid model	Number of solid edges in group $G_i(N)$	Number of concave edges in group $G_i(N_{cv})$	Sum of concave edge angles ($\sum\alpha$)(°)	Angle factor (A)	Factored solid edges ($N+A$)
Unconnected sharp edge			1	0	0	N/A	N/A
Unconnected edge			>= 2	0	0	N/A	N/A
L-Junction			2	1	90	0	2
3-way (T)			2	2	180	1	3
3-way (Y)			3	3	360	0	3
3-way (acute)			3	2	135	0	3
4-way (X)			4	4	360	0	4
4-way (fan)			3	3	180	1	4
4-way (acute)			4	3	135	0	4

Figure 7.4: Topological Similarity Validation (Source: Lockett [27])

Lockett's topological similarity validation technique shows use of geometric quantities angles, proximity groups, etc. which appears inappropriate. Apart from this, the technique appears limited to only simple academic cases.

Table 7.1 outlines the summary of past research contributions in the relevant topological invariants and similarity assessment approaches:

Author	Input	Method	Approach	Advantages	Limitations
Lipson [130]	Sheet Metal Brep	Topological Invariant	topological invariant for sheet metal features	Validating correctness of sheet metal parts	Insufficient for non manifolds
Helen Lockett [132] [133] [27]	Midsurface	Topology Prediction	Proximity groups adjusted by an angle criterion.	Feature recognition using midsurface	Geometry used in topological validation
Sang Hun Lee [134]	Brep	Topology extended	Sheet model is offsetted to thicken	Combined sheet and solid modelling	Detailed transformations missing

Table 7.1: Survey of Topology Invariants and Validation Approaches

Review of the reported works has observed that, topological validation approaches are not prevalent. The ones which use topological criteria appear to be using some geometric criteria as well, making them susceptible to geometric errors. Thus, there is need to devise a fully topology based validation technique, specifically for assessing the output midsurface.

Following sections present the proposed approach for topologically validating the output midsurface.

7.4 Proposed Approach for Topological Validation of Midsurface

This section presents the approach based on the combinatorial topology [135] for determining the validity of a midsurface computed from CAD model of input sheet metal part. It provides transformation equations based purely on the numbers (that's why the term 'combinatorial') of topological entities such as faces, edges, veritces, holes, rings, etc. The proposed approach is formulated in two opposite transformation directions, as stated below:

- **Solid-to-Surface**: This approach involves predicting output midsurface topological entities using input solid's topological entities. So, at first, the topological entities of the input solid are counted. The proposed dimension-reduction transformation equations are then applied to predict the topological entities of the corresponding ideal midsurface. These predicted entities are then compared with the

actual topological entities of the output midsurface. Any mismatch hints at the error. For example, if the actual number of edges are more than the predicted number of edges, then there is likely gap or missing midsurface patch. Section 7.4.2 provides more details on this approach. In addition, the predicted entities are also checked with standard non-manifold equation (Equation 7.4) to check its validity.

- **Surface-to-Solid**: This approach involves predicting the topological entities of the input solid from that of the generated midsurface. So, at first, the topological entities of the output midsurface are counted. The dimension-addition transformation equations are introduced that are applied to predict the topological entities of the corresponding ideal input solid. These predicted entities of ideal input solid are then compared with the actual topological entities of the input solid. Any mismatch hints at the error. For example, the predicted entities of solid may not be a closed volume,which ideally should be the case. Section 7.4.3 provides more details on this approach. In addition, the predicted entities are also checked with standard manifold equation (Equation 7.3) to check its validity as per theoretical basis of solid modeling.

The topological validation method for the midsurface developed in the present research work has been devised for input solids exhibiting sheet metal shape characteristics. Although the validation method mentioned here is derived for the constant thickness solids, it can be extended to thin-walled parts with variable thickness as well.

The following section gives a brief introduction about some of the fundamental concepts.

7.4.1 Theoretical Background

Many sheet metal CAD modelers represent thin-walled solid using data structure called a Boundary Representation (Brep). In Brep, a solid is represented by a set of connected faces. This section provides the characteristics of Brep and its classification into manifold and non-manifold representations. In the present research, the term 'manifold' refers to a solid object which is bound, closed and homeomorphic to a topological sphere (also known as 2-manifold), whereas 'non-manifold' object does not have such restrictions to closure and completeness. The present research uses 'non-manifold' mainly to denote surfaces, unless stated otherwise. Another clarification is that, although the term "surface" is used while describing transformations from and to solids, the more accurate term is "face". Surface is a geometrical entity whereas face is a topological entity. The proposed approach, being related to topology, use of "surface" is theoretically incorrect. But as "midsurface" is a more widely used term than "midface", this chapter uses term "surface" in place of "face" while describing the proposed

transformations.

As mentioned in the previous section, the proposed approach provides two transformation equations, one, from input solid to its corresponding midsurface,and the second, from midsurface to its corresponding solid.

Such transformations can be formulated if there is a common topological formulation representing equations for both solids and surfaces. Following subsections present such common formulation.

7.4.1.1 Boundary Representation (Brep)

Brep is composed of two parts: topology and geometry [135]. Topological entities are shells, faces, edges, vertices, etc. whereas geometric entities are surfaces, curves, points, etc. Topological entities are defined as:

- *shell (s)* is a connected set of *faces*
- *face (f)* is a bounded portion of a *surface*
- *loop (l)* is a circuit of *edges* bounding a *face*
- *half-edges (he)* are used to create a *loop*.
- *edge (e)* is a bounded portion of a *curve*
- *vertex (v)* lies at a *point*.

Validity of the Brep model is checked using the Euler-Poincaré equation.

7.4.1.2 Euler-Poincaré Equation

Euler's equation for polyhedral solids is:

$$v - e + f = 2 \tag{7.1}$$

where, v, e, and f represent the number of vertices, edges and faces respectively [135]. It was discovered by Leonhard Euler in 1752 and was later generalized by Lhuilier [136] as follows:

$$v - e + f = 2 - 2g \tag{7.2}$$

where, g represents genus or holes h or handles (g and h are considered interchangeable in the present research work). Thus, the Euler Poincaré equation for manifold-solids is:

$$v - e + (f - r) = 2(s - h) \tag{7.3}$$

where r represents rings (internal face loop) and s represents shells.

Solids found in the real world have the property that on any point on the boundary, a small enough sphere at that location is split into two pieces, one inside and one outside the object. Non-manifolds do not obey this rule [136]. Weiler ([137]) can be attributed

for the first significant contribution in defining the non-manifold data structure, called *Radial Edge Structure*. Core to this data structure lies the radial cycle, which is an ordered list of faces around an edge. Similar to manifold, equation for non-manifold topology [138] is:

$$v - e + (f - r) = s - h \tag{7.4}$$

Transformation formulation between these two equations, Eqn. 7.3 and Eqn. 7.4, is apparently not straight-forward. The difference between them is only the constant value 2 as the multiplier on the right hand sides of Eqn. 7.3. This constant, which distinguishes between solid and surface is called as a topological invariant, known as Euler's characteristic (χ).

The significance of this topological invariant is that, for given number of topological entities of a shape, such as faces, edges, vertices, it can predict if the shape is solid or a surface. If it is 2 it is solid, if it is 1 it is a surface. Following is an attempt to understand a generic equation encompassing both (solid and surface) dimensionalities in a single equation form.

Brep's topology equation is defined more generically, i.e. for n dimensional object as:

$$\sum_{i=0}^{D} (-1)^i N_i = \sum_{i=0}^{D} (-1)^i \beta_i = \chi \tag{7.5}$$

For dimensions upto 3 ($i = 3$), equation (7.5) simplifies to:

$$N_0 - N_1 + N_2 = \beta_0 - \beta_1 + \beta_2 \tag{7.6}$$

where, Ns are topological entities of the dimension $0, 1$ and 2 respectively and βs are the Betti numbers. β_0, β_1 and β_2 correspond to the number of connected components, holes and cavities, respectively [139]. One of the fundamental property is that two homeomorphic topological spaces will have the same Euler characteristic and Betti numbers.

The Euler characteristic (χ) in terms of Betti numbers, provides a generic invariant for a shape of any dimension. Manifestation of the Betti numbers in different dimensions is different. So, when input solid is transformed into its corresponding midsurface, the interpretation of the Betti number changes from the manifold domain to the non-manifold domain. Following subsection presents manifestation of the Betti numbers for solids and surfaces respectively, which later will be used to formulate the transformations amongst them.

7.4.1.3 Manifold-Solids

Mapping between topological entities of manifold solids from Euler Poincaré Eqn. 7.3 to Betti numbers from Eqn. 7.6 is as follows:

$N_0 = v$: number of vertices

$N_1 = e$: number of edges

$N_2 = (f - r)$: number of faces (f) - additional *artifact* edges corresponding to inner loops (r)

$\beta_0 = s$: number of components or disjoint parts ($shells$)

$\beta_1 = 2h$: number of independent closed curves drawn without splitting. Twice the genus g or h. For Torus, there are two such circles and one genus-hole. ($2h$)

$\beta_2 = s$: number of space regions created by connected surfaces. For an open surface $\beta_0 = 1$ and $\beta_2 = 0$ whereas for closed surface, β_2 is equal to β_0, which is equal to s.

There is direct mapping between manifold solid's topological entities and Betti numbers. Following subsection investigates if that is true with non-manifold equation as well.

7.4.1.4 Non-manifold-Surfaces

Mapping between topological entities of non manifold surface from Euler Poincaré Eqn. 7.4 to Betti numbers from Eqn. 7.6 is as follows:

$N_0 = v$: number of vertices

$N_1 = e$: number of edges

$N_2 = (f - r)$: number of faces (f) - inner loops (r)

$\beta_0 = s$: number of components or disjoint parts ($shells$)

$\beta_1 = h$: number of independent closed curves drawn without splitting. Inner holes (g or h).

$\beta_2 = 0$: number of space regions created by connected surfaces are not present; so it is 0.

Thus there is direct mapping between non manifold surface's topological entities and Betti numbers. Thus Betti numbers provide equivalence between both, manifold and non-manifold equations, and so, transformations between them is feasible.

Apart from Brep representing solids, cellular topology has also been used in the present research work (Section 6.3.3). It has been leveraged in one of the two, transformation directions in the proposed topological validation approach. Following section elaborates its representation and constituents.

7.4.1.5 Cellular Topology

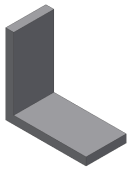
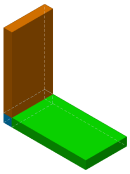
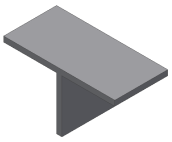
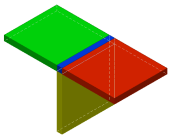
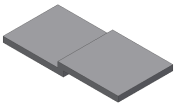
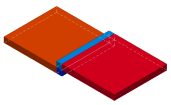
Cellular topology is one of the prominent representations in solid modeling, typically generated by decomposition a solid volume into sub-volumes, called ‘’cells”. According to Chen et al. [119], a cellular model includes topologies of various dimensions.

$$\mathbb{M} = (\cup_{i=1}^{q} C_i^0) \cup (\cup_{j=1}^{r} C_j^1) \cup (\cup_{k=1}^{s} C_k^2) \cup (\cup_{l=1}^{t} C_l^3)$$

where C_i^0 are 0-dimensional vertices, C_j^1 are 1-dimensional edges, C_k^2 are 2-dimensional faces and C_l^3 are 3-dimensional solids.

Table 7.2 shows input solid models along with their corresponding cellular topology models.

Table 7.2: Decomposition of Shapes into Cells

Input Solid	Cellular Model	Input Solid	Cellular Model
			
			

The cells have the following properties:

- **Boundary**: Except C_i^0 cells, all cells are bound by cells with a dimension lower by 1.
- **Overlap**: No cells overlap. $C_i \cap C_j = \phi$
- **Nature**: Either additive or subtractive.

Following sections elaborate the proposed approach of topological validation, in form of two sub approaches. Both sub-approaches take two inputs viz, Brep solid of a sheet metal part’s CAD model and the Brep midsurface. In the first sub-approach, transformation equation of topological entities between the solid and its corresponding midsurface is derived, whereas in the second, transformation equation of topological entities between the midsurface and its corresponding solid is derived.

7.4.2 Solid to Midsurface Transformation

The approach presented below proposes dimension-reduction-transformation equations for predicting the topological entities of its corresponding midsurface. The solid is first decomposed into “Cells”. Cells are denoted as $Cell_{attribute}^{dimension}$. The $_{attribute}$ can

either be the adjacency information i.e. number of neighbor-touching cells, or can be the type of the cell i.e. *h* for hole.

Based on the dimensionality of the cells, they are classified as:

Table 7.3: Classification of Cells

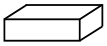
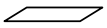
Cell Type	Description	Topological entities count
$Cell^3_*$ 	3D cells (solids), topologically similar to a simple plate	$faces = 6$ $edges = 12$ $vertices = 8$
$Cell^2_*$ 	2D cells, topologically similar to a planar surface	$faces = 1$ $edges = 4$ $vertices = 4$
$Cell^1_*$ 	1D cells, topologically equivalent to a line	$edges = 1$ $vertices = 2$
$Cell^2_h$	Hole is assumed to be cylindrical through-all (true for sheet metal parts)	$edges = 1$ $vertices = 1$

Table 7.3 shows various cell types along with the count of their topological entities. These counts are their contribution to the overall count of topological entities in the formulation developed below. First row shows a 3D cell, of a shape like a plate. Thus it has 6 $faces$, 12 *edges* and 8 *vertices*. The second row, shows a 2D cell, represents rectangular faces, has 1 $face$, 4 *edges* and 4 *vertices*. Similarly other cell types are defined.

The cells are then classified into *sCell*s and *iCell*s as per definitions 3 and 4, respectively. The midsurface cell is of dimension two, so it is represented as $mCell^2$. Figure 7.5 shows "L" shaped solid being decomposed into 3 cells, two of which are *sCell*s and one is *iCell*.

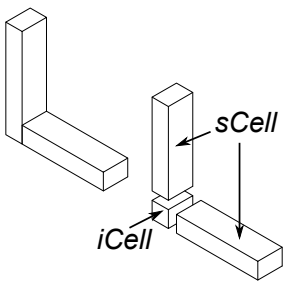
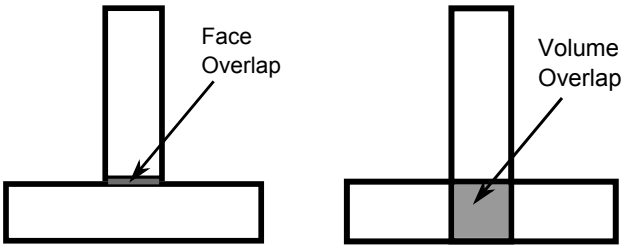


Figure 7.5: Decomposition and Classification of Cells

*iCell*s can be surface interfaces or 2D interfaces, denoted by $iCell^2$ and solid interfaces or 3D interfaces, denoted as $iCell^3$. Figure 7.6a shows surface touching of two

solid cells, also known as 'face overlap' and Figure 7.6b shows volumetric overlap, which after cellular decomposition generates 3D interface cell.



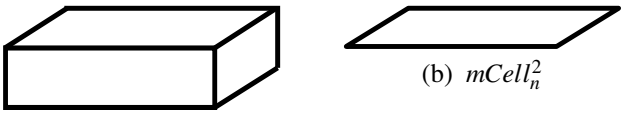
(a) Face Overlap (b) Volume Overlap

Figure 7.6: Face and Volume Interactions of Cells

Topological transformation of solid (3D cells) to its corresponding midsurface (2D cells) is as follows:

- $sCell^3_n$: Solid cell with $adjacency = n$, i.e. n touching sides, transforms into midsurface cell $mCell^2_n$, a surface having n edges. Its topological entities are predicted by following equations as:

$$f = 1; e = 4 - n; v = 4 - 2n \tag{7.7}$$



(a) $sCell^3_n$ (b) $mCell^2_n$

Figure 7.7: Transformation of Solid Cell to Its Midsurface

Figure 7.7 shows a simple plate shaped *sCell* to explain the transformation. This *sCell* has 6 faces, 12 edges and 8 vertices. It is connected with other n cells. When this cell gets transformed to midsurface patch $mCell^2$ then it has entities given by Eq. 7.7. First row of Table 7.4 shows the same case, predicting midsurface entities with $n = 0$, as no other cells are touching this *sCell*.

- $sCell^3_h$: $_h$ denotes 'hole' i.e. a negative solid cell representing. It transforms into midsurface cell $mCell^2_h$, a hole in the surface. Its topological entities are predicted as:

$$e = 1; v = 1 \tag{7.8}$$

- $iCell^3_n$: Interface solid cell with n adjacent touching sides transforms into midsurface cell $mCell_{1,n}$, a edge. Its topological entities are predicted as:

$$e = 1; v = 2 \tag{7.9}$$

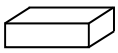
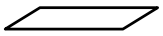
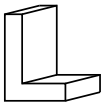
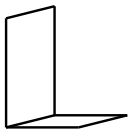
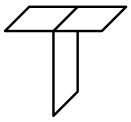
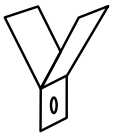
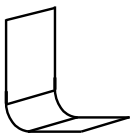
Second row of Table 7.4, i.e. "L" case, has 3 cells, 2 $sCell_0^3$s and one $iCell_2^3$. For the interface cell $iCell_2^3$, $n = 2$, as it is touching 2 solid cells. It will get transformed into an edge with predicted entities as per Eq. 7.9.

- $iCell_2^2$: Interface face cell touched from both sides transforms into midsurface cell $mCell_2^1$, a edge. Its topological entities are predicted as:

$$e = 1; v = 2 \tag{7.10}$$

Table 7.4 lists various basic shapes and their dimension-reduction-transformations into their corresponding midsurface

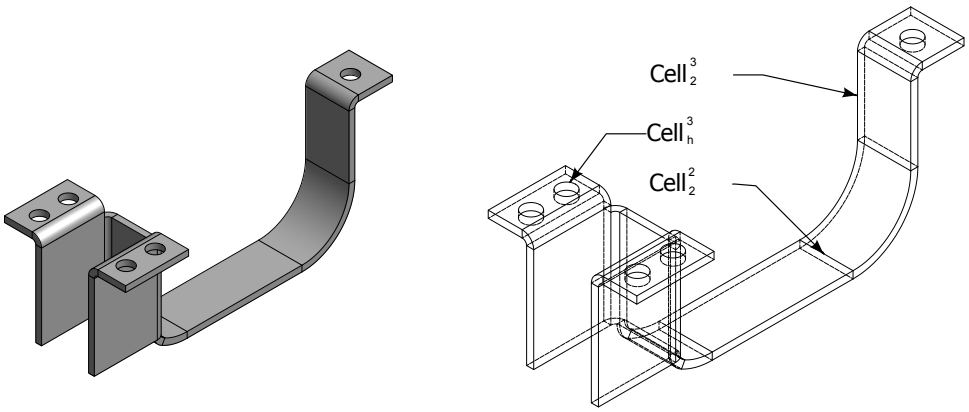
Table 7.4: Topological Validation of Midsurface by Dimensional Reduction

Solid	**Midsurface**	**Solid Cells**	**Midsurface Cells**	**Predicted Topological Entities**
		$sCell_0^3$	$mCell_0^2$	$1f + (4-0)e + (4-2\times 0)v$ $= 1f + 4e + 4v$
		$2 \times sCell_1^3$ $+ iCell_2^3$	$2 \times mCell_1^2$ $+ mCell_2^1$	$2 \times (1f + (4-1)e + (4-2\times 1)v)$ $+ (1e + 2v)$ $= 2f + 7e + 6v$
		$3 \times sCell_1^3 +$ $iCell_3^3$	$3 \times mCell_1^2 +$ $mCell_3^1$	$3 \times (1f + (4-1)e + (4-2\times 1)v) + (1e + 2v) = 3f + 10e + 8v$
		$3 \times sCell_1^3 +$ $iCell_3^3 +$ $sCell_h^3$	$3 \times mCell_1^2 +$ $mCell_3^1 +$ $mCell_h^2$	$3 \times (1f + (4-1)e + (4-2\times 1)v) + (1e + 2v) + (1e + 1v) = 3f + 11e + 9v$
		$2 \times sCell_1^3 +$ $2 \times iCell_2^2 +$ $sCell_2^3$	$2 \times mCell_1^2 +$ $2 \times mCell_2^1 +$ $mCell_2^2$	$2 \times (1f + (4-1)e + (4-2\times 1)v) + 2 \times (1e + 2v) + (1f + (4-2)e + (4-2\times 2)v) =$ $3f + 10e + 8v$

One sample case is explained here. The third row, case "T", shows solid model and corresponding midsurface in first two columns. Cellular decomposition of the solid results in 3 solid cells $sCell_1^3$, each with $n = 1$, i.e. 1 adjacent cell and one interface cell $iCell_3^3$. So the solid is represented $3 \times sCell_1^3 + iCell_3^3$. Dimension reduction of these cell, reduces the dimensionality by 1, and thus result in midsurface equation $2 \times mCell_1^2$

$+ mCell_2^1$. Each of these transformations contribute topological entities as per Eqn. 7.7 and Eqn. 7.9, and result in predicted midsurface entities as $3 \times (1f + (4-1)e + (4 - 2 \times 1)v) + (1e + 2v) = 3f + 10e + 8v$ i.e. 3 faces, 10 edges and 8 vertices.

Figure 7.8 shows working of the above mentioned approach on a relatively-complex practical shape.



(a) Input Solid Model (b) Cellular Decomposition

Figure 7.8: Solid to Surface Transformation Approach for Bracket

Figure 7.8a shows the input solid whereas Figure 7.8b shows the cellular topology. Steps listed below show formulations as per the equations derived before.

- **Solid cells**:
  $5 \times sCell_{3,h} + 3 \times sCell_1^3 + 13 \times sCell_2^3 + 14 \times iCell_2^2$
- **Transformed Midsurface Cells**:
  $5 \times mCell_h^2 + 3 \times mCell_{2,1} + 13 \times mCell_2^2 + 14 \times mCell_2^1$
- **Predicted midsurface Entities**:
  $5(1e + 1v) + 3(1f + 3e + 2v) + 13(1f + 2e + 0v) + 14(1e + 2v) = 16f + 54e + 39v$

The derived formulation (Eqn 7.9, 7.10, 7.7, 7.8) predicts correct topological entities for the midsurface. These, when substituted in the non-manifold equation (Eqn 7.4) also prove to be valid. With $s = 1, r = 5, h = 5$, the equation matches both sides: $39 - 54 + (16 - 5) = 1(1 - 5)$

.

Thus the above mentioned approach can be used to predict midsurface entities of an input solid. For validation, these predicted entities need to be compared with the topological entities of the actual midsurface. If they match, then topological validation is successful and then geometric validation can be carried out. If there is any mismatch, the errors need to investigated and fixed. Unlike the approach presented by Lockett [27], the proposed approach elaborated above, uses only topological entities and not the geometrical ones like angles, proximity groups, etc. as done in Lockett's work.

Following subsection presents the second sub-approach, that is of dimension-addition.

7.4.3 Midsurface to Sheet Metal Solid Transformation

In this approach, given a midsurface, topological entities of its corresponding sheet metal solid are predicted. These predicted entities are verified to check if they match with the topological entities of the actual solid. Apart from this, the predicted entities are also validated against the manifold equation (Eqn 7.3).

The proposed approach here uses additional classification of topological entities, that is typically available in Brep. These additionally classified entities are used to propose the dimension addition transformation. For example, vertices are classified as sharp, radial, internal, whereas edges are classified as sharp,cross-radial, side-radial, internal, etc. The proposed approach provides dimension addition of these entities as they get transformed into topological entities of the solid.

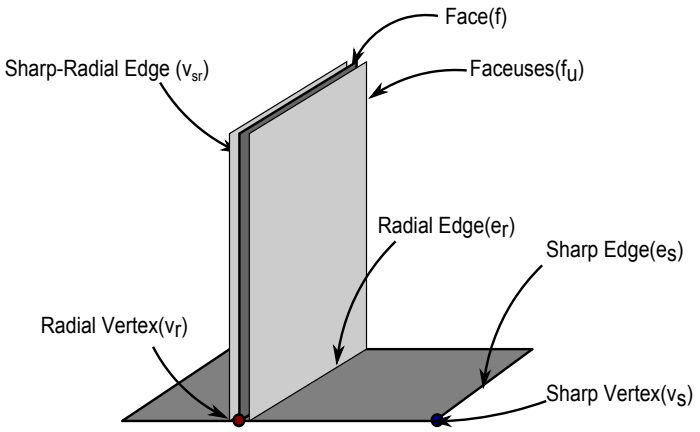


Figure 7.9: Non-Manifold Topological Entities

Figure 7.9 shows midsurface of "T" shaped solid, along with the classified topological entities. Apart from known topological entites, a few new ones have been defined and used in the approach, as below:

- Face (f): A face is composed of two face-uses f_u. Face-uses are also known as Co-Faces or Half-Faces in other Brep terminologies [135].
- Sharp Vertex (v_s): A vertex is a sharp vertex if it is connected to two edges of the same face.
- Sharp Edge (e_s): An edge is a sharp edge if it is connected to two sharp vertices.
- Radial Vertex (v_r): A vertex is a radial vertex if it is connected edges of different faces.
- Degree (n_r) at the radial edge is the number of faces attached to it
- Cross Radial Edge (e_r): An edge is a cross radial edge if it is connected between two radial vertices and connects two different faces.
- Side-Radial Edge (e_{rr}): An edge is a side radial edge if it is connected between two radial vertices and is of same face

- Sharp-Radial Edge (e_{sr}): An edge is a sharp radial edge if it is between sharp and radial vertex.
- Internal Edge (e_i): An edge is an internal edge if it is part of the inner edge-loop or ring of the face.
- Internal Vertex (v_i): A vertex is an internal vertex if it is connected to the internal edge.
- Internal Loop (r_i): A loop is an inner loop if it is composed of internal edges and vertices.

Using the topological entities defined above, the dimension addition transformation is elaborated as follows. The basic principal of the approach is that a solid can be imagined to be a thickened midsurface. Each topological entity of the midsurface can be transformed to its higher-dimension counter part to create a solid. The solid will be of thin-walled type and will have thickness faces, called capping faces as well as non-thickness faces, called principal faces.

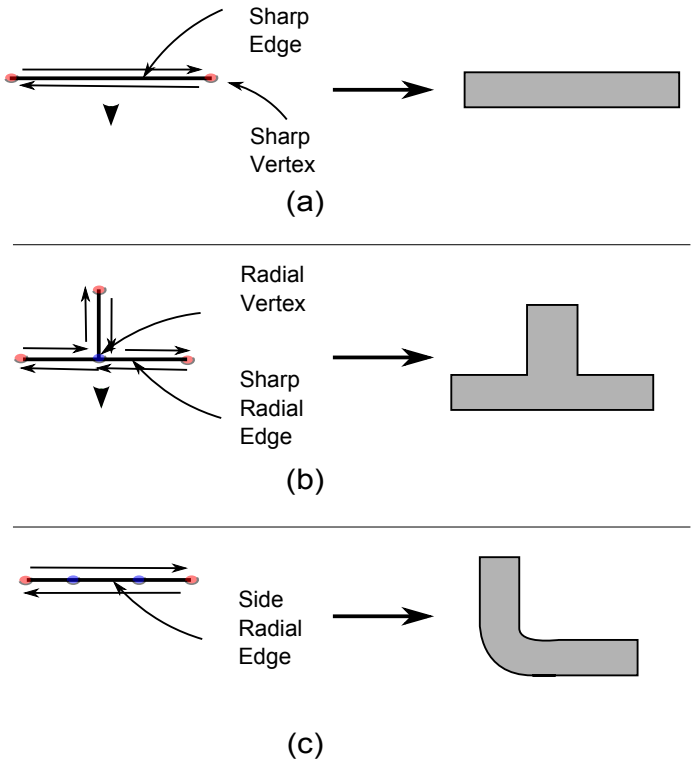


Figure 7.10: Transformation of Loops from Midsurface to Faces of the Solid

Figure 7.10 shows some of the dimension addition transformations, from midsurface topological entities to solid topological entities. Figure 7.10(a) shows that an edge on midsurface, which has got classified as a sharp edge, becomes a face in the solid. Figure 7.10(b) a "T" Shaped junction of edges, which have got classified as radial edges, along with a radial vertex, get transformed into a face in the solid.

The topological entities of the generated solid are transformed from midsurface entities as per following rules:

- Face-uses on the midsurface become principal faces in the solid.
- Sharp vertices on the midsurface become capping edges in the solid.
- Apart from edge-use loop corresponding to face-use, a new loop is proposed for side-capping faces. The loop is formed between two sharp vertices (v_s) using more than one sharp (e_{sr}) or side radial (e_{rr}) edges but not using the cross radial edge (e_r). Such independent paths creating individual side faces are called l_p.
- Loop between two sharp vertices. This gives rise to a singular capping face (Fig. 7.10 a).
- Loop between three branched sharp vertices. This gives rise to a combined capping face (Fig. 7.10 b).
- Loop between two sharp vertices with multiple radial vertices in between them. This gives rise to a combined capping face (Fig. 7.10 c).

Based on the rules above, classified topological entities on the midsurface contribute to topological entities of the solid. Conversely, the topological entities of solid are made up of contribution from dimension addition of various topological entities on the midsurface. Thus, the proposed approach predicts topological entities in the solid with formulations as follows. The term "manifold" and the corresponding suffix $_m$, are used for predicted entities of the solid.

- **Manifold-Vertices (v_m)**: Number of vertices in the solid are calculated by doubling the number of sharp and internal vertices (one up and one below) + vertices for junctions which are denoted by the summation of number of radial vertices times their corresponding degrees.

$$v_m = 2(v_s + v_i) + \sum n_r v_r \tag{7.11}$$

- **Manifold-Edges (e_m)**: Number of edges in the solid are calculated by doubling the number of sharp, sharp-radial and internal edges (offset up and down) + degree times radial edges for offsets at junctions + sharp vertices for vertical-capping edges + internal vertices for vertical seam edges.

$$e_m = 2(e_s + e_{sr} + + e_{rr} + e_i) + \sum n_r e_r + v_s + v_i \tag{7.12}$$

- **Manifold-Faces (f_m)**: Number of faces in the solid are calculated by doubling the number of faces (offset up and down) + sharp edges for capping faces + paths to have one combined face + internal edges for capping internal faces

$$f_m = 2f + e_s + l_p + e_i \tag{7.13}$$

- **Manifold-Shells (s_m)**: Number of shells remain the same in the solid.

- **Manifold-Rings (r_m)**: Number of rings in the solid are calculated by doubling the number of internal rings on the midsurface.

$$r_m = 2r_i \tag{7.14}$$

- **Manifold-Genus (h_m)**: Number of Internal ring on midsurface become same number of holes (genus) in the solid.

$$h_m = r_i \tag{7.15}$$

Once the manifold (i.e solid) topological entities are calculated, they can be verified by Euler-Poincaré Equation (Eqn 7.3) by comparing left and right hand sides of the equations. Each side has to match to an invariant denoted as χ. Similarly input midsurface's verification can be done by Equation (Eqn 7.4).

With the above mentioned transformation rules, the proposed approach states procedure to validate the midsurface as below:

1. The number of classified topological entities of the midsurface are counted as per the definitions suggested in Section 7.4.3 and shown in Figure 7.9

2. Based on dimension addition transformation equations 7.11, 7.12, 7.13, 7.14, 7.15, number of predicted topological entities of the corresponding input solid are calculated using equations, as follows:

   (a) Predicted number of manifold faces: f_m
   $= 2f + e_s + l_p + e_i$

   (b) Predicted number of manifold edges: e_m
   $= 2(e_s + e_{sr} + e_{rr} + e_i) + \sum n_r e_r + v_s + v_i$

   (c) Predicted number of manifold vertices: v_m
   $= 2v_s + \sum n_r v_r + 2v_i$

   (d) Predicted number of manifold shells, holes and rings:
   $s_m = s = 1, h_m = r_i = 0, r_m = 2r_i = 0$

3. Verify that the topological entities of the midsurface satisfy the non-manifold equation (Equation 7.4), by deducing that the left (χ_{nml}) and right (χ_{nmr}) hand side of the equation matches.

   (a) Input midsurface's non-manifold equation's left side χ_{nml}
   $= v - e + f$

   (b) Input midsurface's non-manifold equation's right side χ_{nmr}
   $= s - h + r$

4. Verify that the predicted topological entities of the thin-walled solid satisfy the manifold equation (Equation 7.3), by deducing that the left (χ_{ml}) and right (χ_{mr}) hand side of the equation match; thus proving that the transformation equations are valid.

   (a) Solid's manifold equation's left side χ_{ml}
   $= v_m - e_m + f_m$

   (b) Solid's manifold equation's right side χ_{mr}
   $= 2(s_m - h_m) + r_m$

Table 7.5 displays the validation of midsurface using proposed dimension-addition-transformation equations. It is evident that the derived formulation works as the predicted entities validate the topological invariant χ_{m*} correctly. The validation is complete when these entities match with the actual solid's topological entities, and geometrical validation also assesses it positively.

Table 7.5: Validation of Midsurface (M)

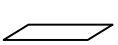
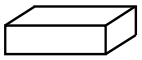
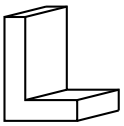
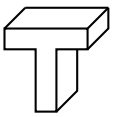
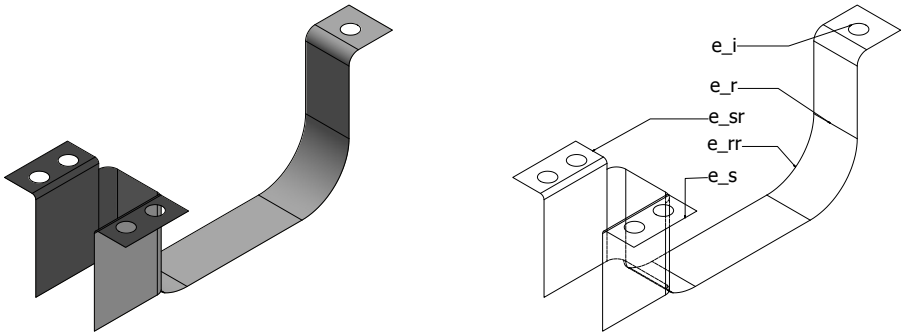
M	f	l_p	e_s	e_{sr}	e_r	e_{rr}	e_i	v_s	v_r	v_i	f_m	e_m	v_m	χ_{m*}	**Solid**
	1	0	4	0	0	0	0	4	0	0	6	12	8	2	
	2	2	2	4	1	0	0	4	2	0	8	18	12	2	
	3	2	3	6	1	0	0	6	2	0	11	27	18	2	
	3	2	3	6	1	0	1	6	2	1	12	30	20	2	
	3	2	2	4	2	2	0	4	4	0	10	24	16	2	

Figure 7.11a shows the midsurface of a sheet metal bracket's solid model, whereas Figure 7.11b shows classified topological entities of the midsurface.

Steps listed below show formulations as per the equations derived before.

- **Midsurface entities**:
  $f = 15, e_s = 3, e_{sr} = 10, e_r = 14, e_{rr} = 19, l_p = 9, e_i = 5, v_s = 8, v_r = 24, v_i = 5, s = 1, h = 5, r = 5$



(a) Input Midsurface (b) Topological Entities Classification

Figure 7.11: Solid to Surface Transformation Approach for Bracket

- **Predicted number of manifold-faces**:
  $f_m = 2f + e_s + l_p + e_i$
  $= 2 \times 15 + 3 + 9 + 5 = 47$
- **Predicted number of manifold-edges**:
  $e_m = 2(e_s + e_{sr} + e_{rr} + e_i) + \sum n_r e_r + v_s + v_i$
  $= 2(3 + 10 + 19 + 5) + (2 \times 12 + 4 \times 2) + 8 + 5 = 119$
- **Predicted number of manifold-vertices**:
  $v_m = 2(v_s + v_i) + \sum n_r v_r$
  $= 2 \times (8 + 5) + 2 \times 24 = 74$
- **Predicted number of manifold-shells-holes**:
  $s_m = s = 1, h_m = r_i = 5, r_m = 2r_i = 10$
- **Input midsurface's non-manifold equation's left side**: χ_{nml}
  $= v - e + f$
  $= 32 - 46 + 15 = 1$
- **Input midsurface's non-manifold equation's right side**: χ_{nmr}
  $= s - h + r$
  $= 1 - 5 + 5 = 1$
- **Solid's manifold equation's left side**: χ_{ml}
  $= v_m - e_m + f_m$
  $= 74 - 119 + 47 = 2$
- **Solid's manifold equation's right side**: χ_{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 ($\chi_{ml} = \chi_{mr} = 2$). Validation can also be performed by comparing the topological entities of the thin-walled solid with the predicted ones.

7.5 Conclusions

This chapter presented the approach of topological validation of midsurface, computed from sheet metal part model. It has two phases. In the first phase, called "solid to surface", given the topological entities of the input solid, its midsurface entities are predicted. In the second, called "surface to solid", given the topological entities of midsurface, the topological entities of its corresponding input solid are predicted. Any mismatch between predicted entities and the actual ones, suggests error, which needs to be located and fixed. Shape with or without draft angle are topologically the same, so the formulation developed here applies to them equivalently.

The proposed approach scores over the past topological validation approaches in the fact that it is purely topologically based and does not use any geometrical entities. It is not heuristic in nature and is less error prone. the topological validation method elaborated here should be used in conjunction with the existing geometric validation method, for completeness. The examples demonstrated show efficacy of the proposed approach not only for simple academic models but also for the real-life sheet metal part models.

Chapter 8

Case-studies

8.1 Introduction

The objective of this chapter is to present case studies that demonstrate capabilities of **MidAS** in an integrated and top-down manner. It initially demonstrates two examples with a detailed explanation of each step and then shows a few more industrial sheet metal part models in a summary format.

Computation of midsurface depends to a great extent on factors such as the variety of sheet metal features used, dependencies amongst them, complexity of geometries of curves and surfaces, etc. In **MidAS**, these factors directly affect working of various sub-modules such as defeaturing, generalization, etc. In order to test capabilities of **MidAS**, a number of industrial sheet metal part models are selected and tested. The first two case studies focus on following capabilities of **MidAS**:

- Generating gross shape of the input sheet metal part model by defeaturing.
- Generalization of remaining features into Loft-equivalent features in $\mathcal{ABLE}$ model.
- Feature based Cellular decomposition of $\mathcal{ABLE}$ model.
- Computing midsurface using Cellular Topology.
- Reapplication of Dormant features tool bodies on the midsurface.
- Using the output midsurface for CAE analysis.

8.2 Case Study I

The test part chosen for this case study is “Enclosure”, which is a typical sheet metal casing model used in electronics equipments. It houses circuits, wires, fans, etc. The CAD feature model shows outer casing, two flaps with holes for screw fitments. It has slots for interfaces to the external environment. Some superficial features include embossed name, and array of slots for guiding wires in place. A chute on one side is to keep bus-wires in place.

8.2.1 Input CAD Model

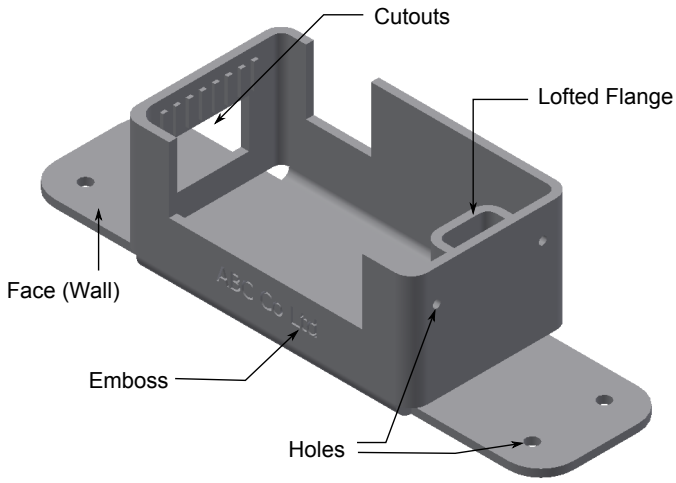


Figure 8.1: Input Sheet Metal CAD Model

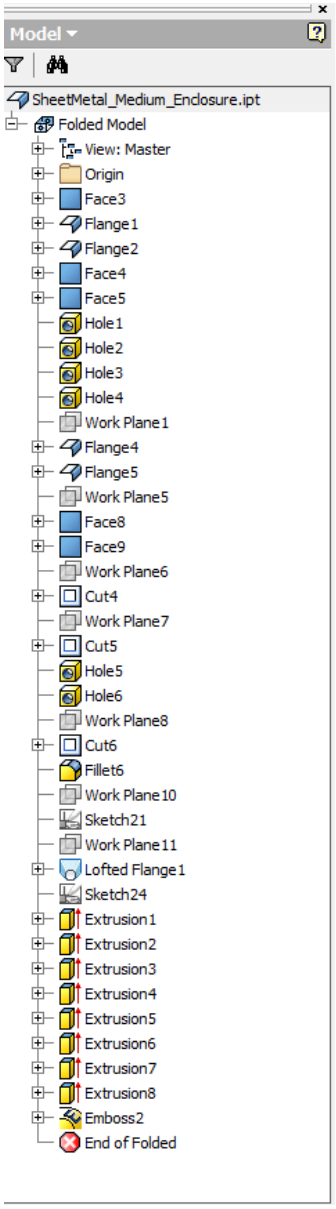


Figure 8.2: Feature Tree

Input to **MidAS** has been modeled using Sheet Metal modeling environment of Autodesk Inventor. Figure 8.1 shows the model and Figure 8.2 shows the corresponding feature tree.

The model has 3 cutouts for components interfacing with outside world, letter embossing, a chute for wires and holes for fixing bolts. Sheet metal features such as Face (Wall), Flange, Hole, Lofted Flange, Emboss, etc. have been used. Dependencies amongst features was minimized by not referencing faces or edges of previously modeled features, but by using reference geometries such as planes, axes, etc.

8.2.2 CAD Model Defeaturing

Defeaturing removes irrelevant features to compute the "gross shape". It also caches tool-bodies of relevant negative features to be used for piercing after midsurface computation. Threshold (D) used as a size threshold here is 5% of the total model size, computed using face-area-summation method.

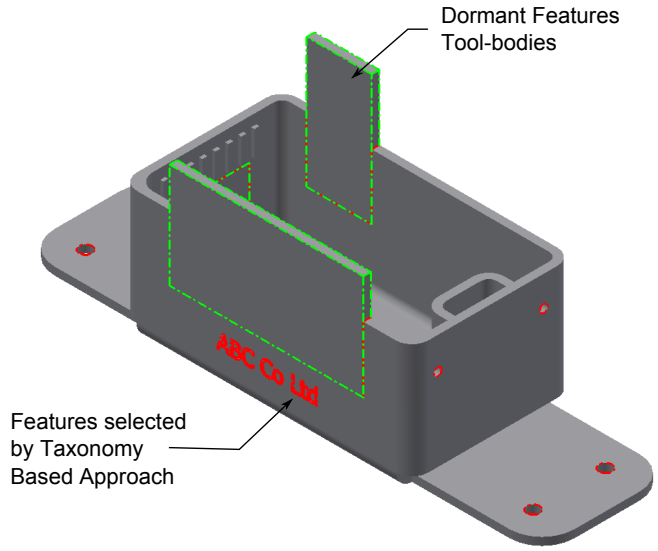


Figure 8.3: Selection of Features based on Taxonomy Based and Dormant Features Approach

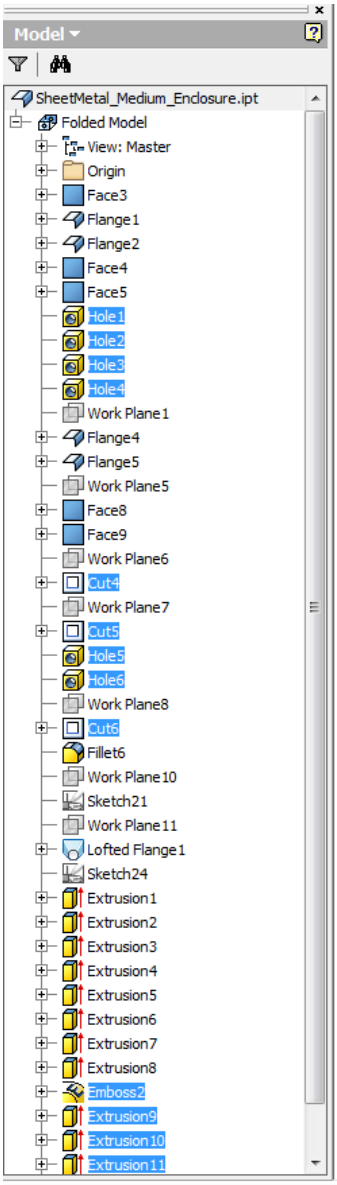


Figure 8.4: Selected Features for Removal

As per defeaturing based on Sheet Metal feature taxonomy rules, elaborated in Section 4.4.1, secondary features chosen for removal are:

- Hole1, Hole2, Hole3, Hole4, Hole5, Hole6
- Cut4, Cut5, Cut6
- Emboss2

Figure 8.3 also shows the identification of dormant features. As elaborated in Section 4.6, relevant negative features are removed after storing their tool-bodies. These tool-bodies, represented by Extrusion9, Extrusion10 and Extrusion11, are shown in Figure 8.3 as "Dormant Features". Figure 8.4 shows the same features selected in the feature tree.

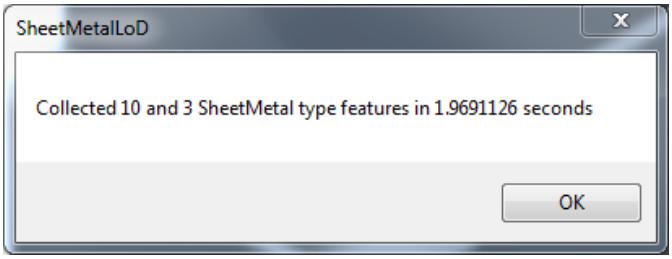


Figure 8.5: Phase I and III Selection Timings

Fig. 8.5 shows time taken for detection of suppressible sheet metal and dormant features.

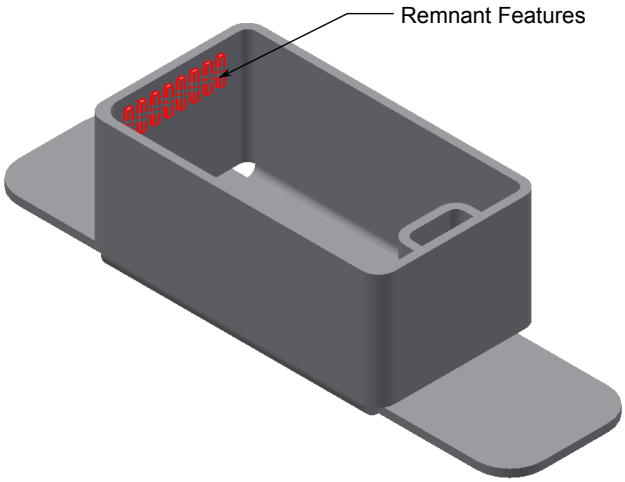


Figure 8.6: Selection of Features based on Remnant Features Approach

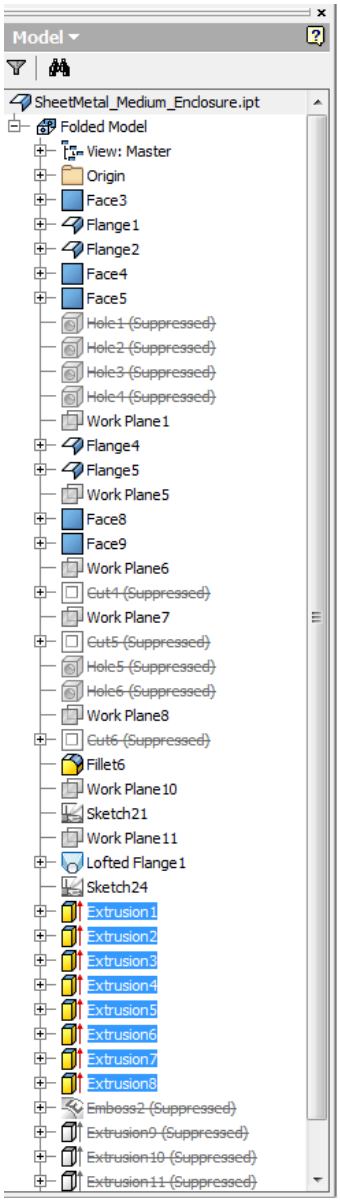


Figure 8.7: Selected Features for Removal

As per defeaturing based on Remnant feature portions, elaborated in Section 4.5, Figure 8.6 shows Extrusion1 to 8 getting selected and removed. They were part of array of small protrusions. Their remnant portions were below the threshold size, thus got selected as seen in Figure 8.7.

Figure 8.9 shows the result of defeaturing process whereas Figure 8.10 shows the feature tree of the same model. The model has been simplified substantially while retaining the gross shape intent.

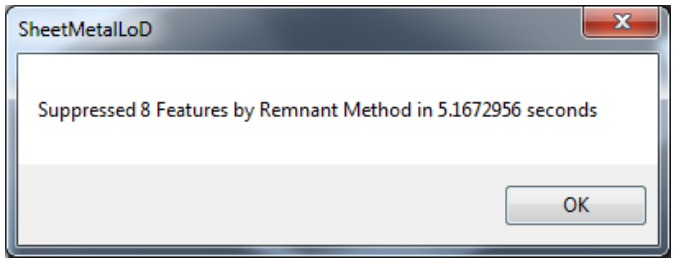


Figure 8.8: Phase II Selection Timings

Fig. 8.8 shows time taken for detection of suppressible features based on Remnant Features method.

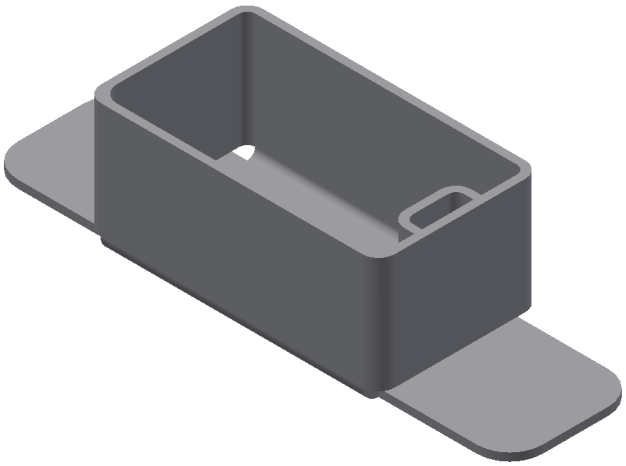


Figure 8.9: Gross Shape after Full Defeaturing

Based on Eqn. 4.1. Defeaturing Effectiveness is computed as follows:

Entity	Input	After Phase I	Output
Faces	259	104	64
Features		13	8

$$pR = (1 - \frac{64}{259}) \times 100 = 75.29\%$$

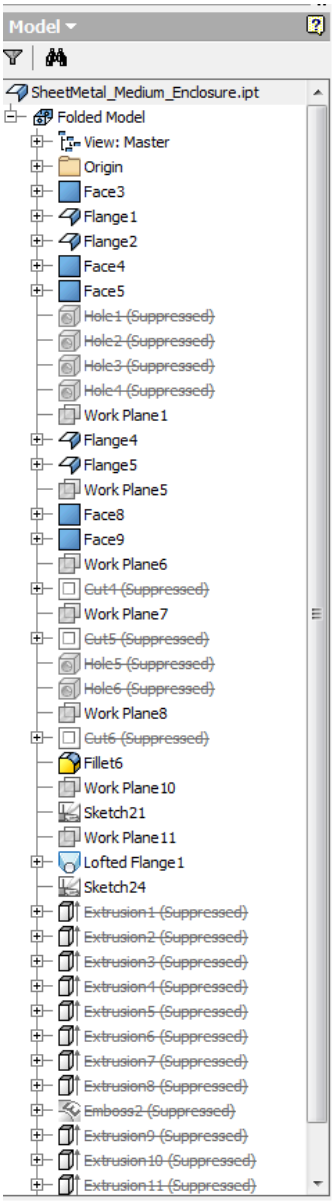
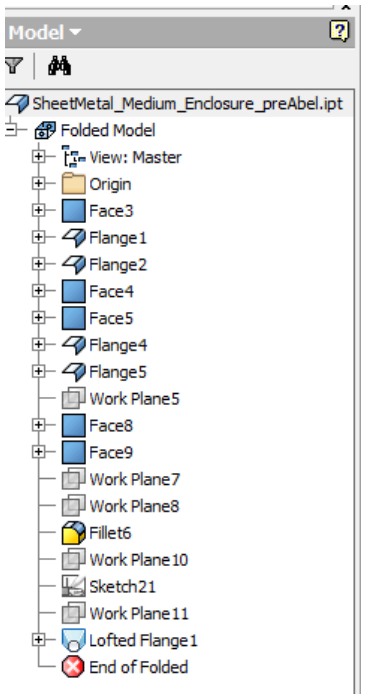


Figure 8.10: Feature Tree

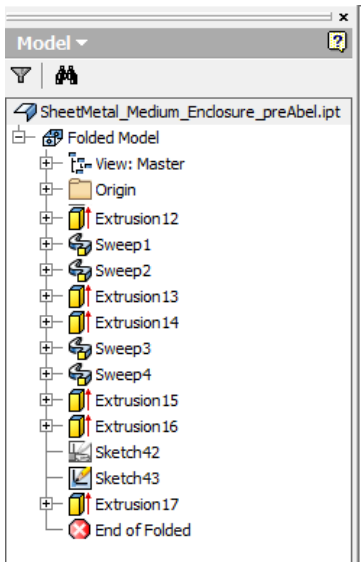
The Table shows number of faces in the CAD model, at the Input stage, then after application context specific defeaturing (called "Phase I") and then at the final stage (called "Output"). So, overall, the number of faces have reduced from 259 in the input to 64 in the output model. The number of features left after Phase I were 13 whereas 8 were remaining at the end. It can be clearly seen that even after 75% reduction in number of faces the gross shape of the enclosure retains the overall shape and design intent.

8.2.3 CAD Model Generalization

The purpose of this module is to transform remaining sheet metal features (Figure 8.11a) into $\mathcal{ABLE}$ features (Figure 8.11b).



(a) CAD Feature Tree



(b) $\mathcal{ABLE}$ Feature tree

Figure 8.11: Transformation of CAD Model to Generalized Features

Table 8.2 shows map of Sheet Metal features to their corresponding $\mathcal{ABLE}$ such as Extrude, Sweep, etc. as per approach detailed in Section 5.4.6.

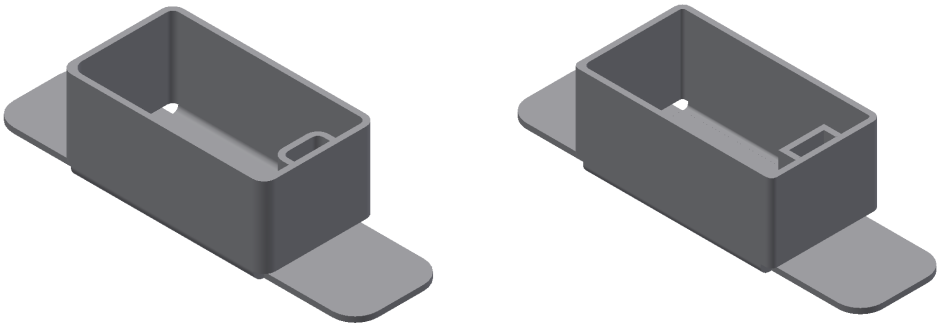
Table 8.2: Case I: CAD to $\mathcal{ABLE}$ Feature Mapping

CAD Feature	$\mathcal{ABLE}$ Feature
Face3	Extrusion12
Flange1	Sweep1
Flange2	Sweep2
Face4	Extrusion13
Face5	Extrusion15
Flange4	Sweep3
Flange5	Sweep4
Face8	Extrusion15
Face9	Extrusion16
LoftedFlange1	Extrusion17

Figure 8.12a shows the input model having sheet metal features such as Face-Wall, Flange, Lofted Flange, etc. Figure 8.12b shows the transformed feature tree. Algorithms for transforming each of these features are listed in Table 5.2.

Fig. 8.13 shows time taken for transforming CAD Sheet Metal features to $\mathcal{ABLE}$ features.

Thus, after transforming all the sheet metal features in the input model to their $\mathcal{ABLE}$ features, the output $\mathcal{ABLE}$ model retains the overall shape and design intent.



(a) Input CAD Model (b) Transformed $\mathcal{ABLE}$ Model

Figure 8.12: Transformation of CAD Model to $\mathcal{ABLE}$ Model

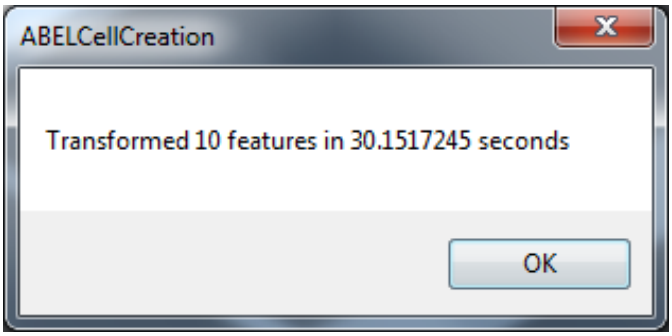


Figure 8.13: $\mathcal{ABLE}$ Transformation Timings

8.2.4 CAD Model Decomposition

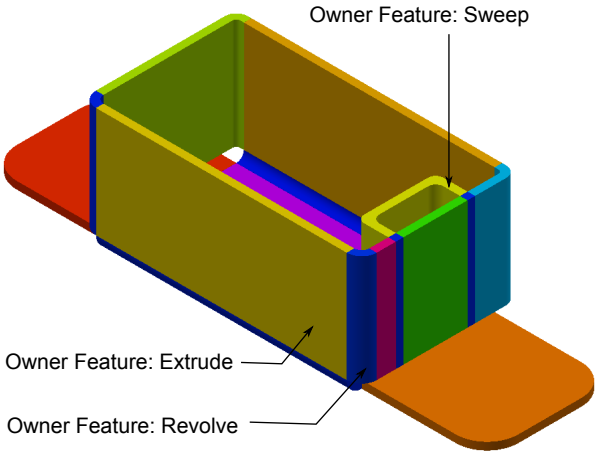
In the present research work the $\mathcal{ABLE}$ model is appropriately decomposed manually to create solid and interface cells. As elaborated in Section 6.3.2, the decomposition is done in two steps viz. Feature partitioning and convex partitioning. In feature partitioning "Unite" booleans are set to "New Body" type thereby separating the tool bodies of features. Then feature volumes are split at the concave edges, by Convex Partitioning. Figure 8.14a shows the decomposed model and Figure 8.14b shows the feature tree.

Thus, the $\mathcal{ABLE}$ model is now transformed into a collection of cells with their respective owner features.

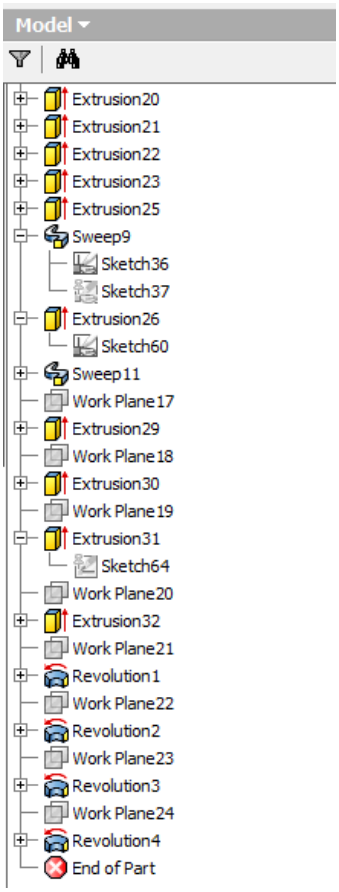
8.2.5 Midsurface Generation

With decomposition, the complex problem of computing midsurface of a model is reduced to a set of more manageable and deterministic sub-problems, i.e. cell-wise midsurface computation. The decomposed model has 14 *sCell*s and 2 *iCell*s.

In the first step, all *sCell*s compute their own midsurface patches. Figure 8.15a shows an example *sCell*. Its sketch-profile is extracted, and facet-ed as shown in Fig-



(a) Cells with $\mathcal{ABLE}$ owner features



(b) Feature Tree After Decomposition

Figure 8.14: Cellular Decomposition of the Gross Shape

ure 8.15b. Polygon is decomposed as depicted in Figure 8.15c and Figure 8.15d shows the output midcurve, which is extruded to compute the midsurface patch.

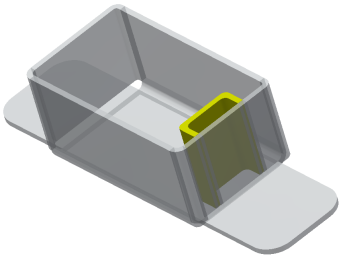
Figure 8.16a shows the output of this step.Figure 8.16b shows magnified portion where midsurface patches are yet to be joined.

In the next step, midsurface pacthes are joined together at the *iCell*s.

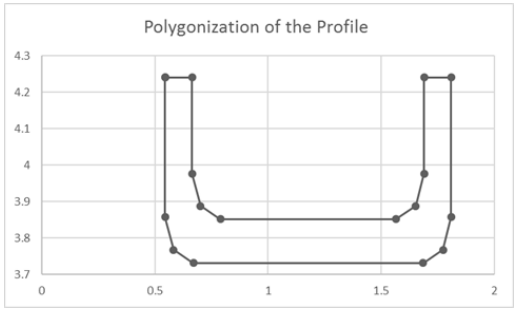
Figure 8.17 shows working of the midsurface patch joining approach. Figure 8.17a depicts subset of the model. It shows two interface (*iCell*) cells, attached in two "T" type junctions. Cells with owner features f_3 and f_5 are the *iCell*s, and rest are the *sCell*s. Figure 8.17b shows, schematically, how extensions are performed with a generic rule specified in Section 6.5.

Figure 8.18 shows midsurface patches and their connections resulting into a well connected midsurface.

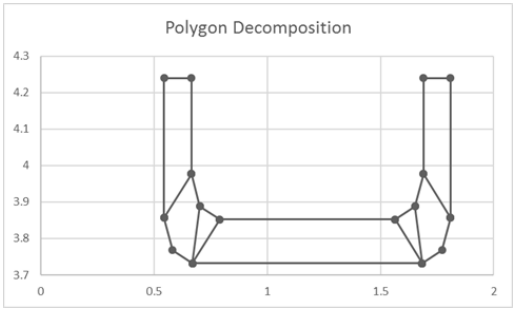
Fig. 8.19 shows time taken for computing midsurface including computation of midcurves, patches and their joining.



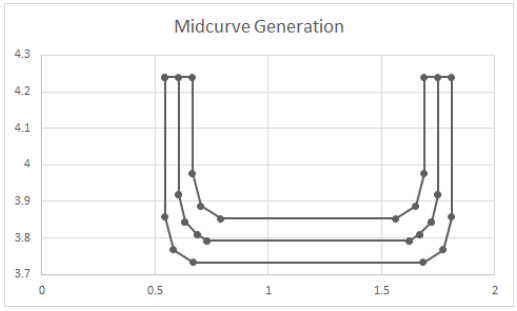
(a) Example Solid Cell



(b) Profile Polygon

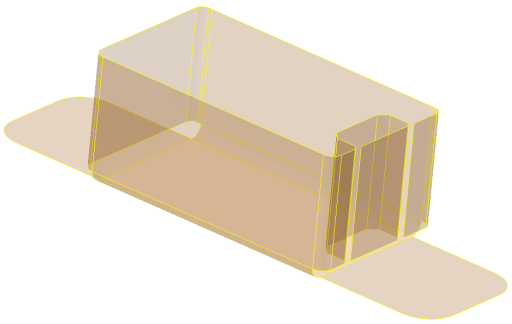


(c) Polygon Decomposition

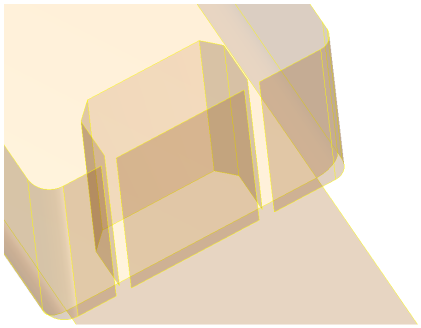


(d) Resultant Midcurve

Figure 8.15: Midcurve Computation of a Solid Cell

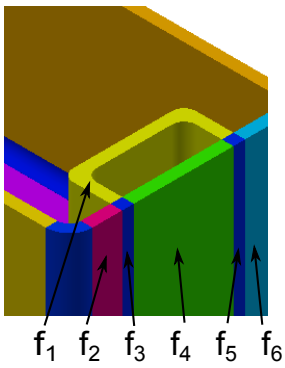


(a) Midsurface Patches from all Solid Cells

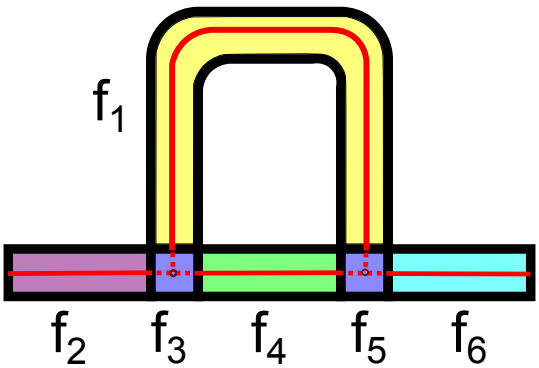


(b) Gaps at Interface Cell Locations

Figure 8.16: Midsurface Patches at *sCell*s



(a) Interface Cells f_3, f_5



(b) Connections at Interface Cells

Figure 8.17: Interface Cells Connecting Midsurface Patches

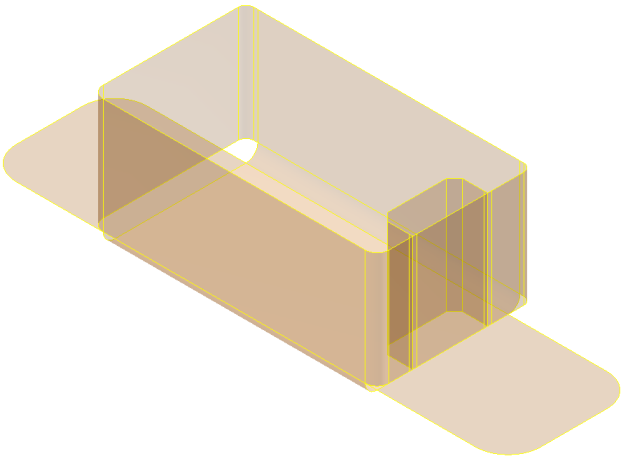


Figure 8.18: Connected Midsurface After Processing of All Solid and Interface Cells

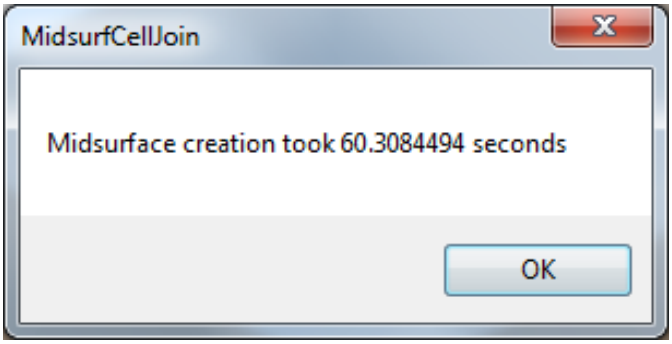


Figure 8.19: Midsurface Computation Timings

Use of Cellular topology has removed need of face-pairing approach altogether. It has simplified the problem domain considerably. Thus the midsurface computation has become more generic and agnostic to complexity of the input model.

8.2.6 Dormant Feature Re-application

Midsurface generated at this stage is not final yet. As elaborated in Section 4.4.1, during defeaturing, negative features are removed even though they are relevant. The intent is to make midsurface computation simpler. These temporarily removed features, called Dormant features, are reinstated back on the midsurface here.

Cached dormant feature tool bodies are pierced into the midsurface to restore the temporarily-suppressed negative features. Figure 8.20 shows the tool-bodies being re-applied on the midsurface, whereas Figure 8.21 shows the output final midsurface.

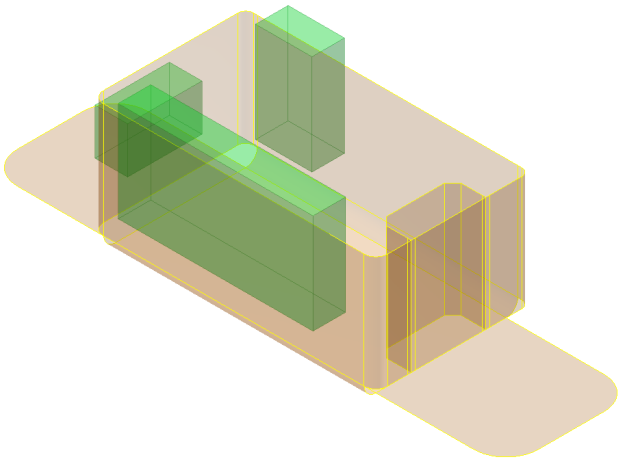


Figure 8.20: Re-application of Dormant Feature Tool-bodies on the Connected Midsurface

8.2.7 Final Output

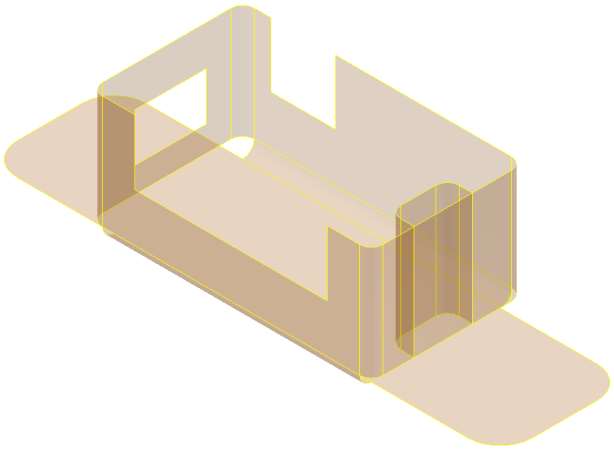


Figure 8.21: Output Midsurface computed by **MidAS**

Figure 8.21 shows the final midsurface output computed by **MidAS**. It has well-connected midsurface, does not have irrelevant features and has maintained the flow of input model shape with fidelity. It is the desired output.

In order to compare the output midsurface, its benchmarking is done against the output by a commercial system. Figure 8.22 shows the midsurface computed by a commercial CAD-CAE system.

There are few critical problems in the output midsurface computed by a commercial CAD-CAE system. At two places, the midsurface patches are missing. The lettering (i.e

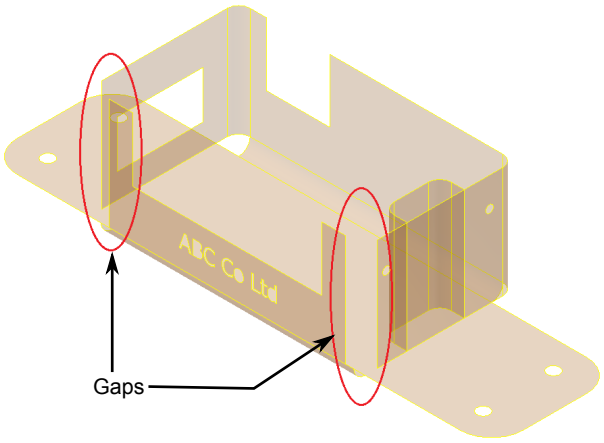


Figure 8.22: Midsurface computed by a Commercial System

Embossing) has several problems. The Emboss should have been removed altogether.

Following are some of the salient conclusions with regard to the test case elaborated above:

- Defeaturing module reduced complexity of the input model substantially (about 75% reduction in number of faces) and still retained the gross shape.
- Generalization module reduced variety of sheet metal to a limited set of $\mathcal{ABLE}$ features.
- Decomposition module reduced the complexity further by dividing the model in to manageable set of cells.
- Midsurface was computed from the cells using generic rules, compared to some of the existing approaches where case-to-case heuristics has to be applied.
- Compared to the output by a commercial CAD-CAE system, the midsurface computed by **MidAS** clearly has a better output.

Following sections demonstrate efficacy of **MidAS** using few other real-life sheet metal part models.

8.3 Case Study II

This case study tests **MidAS** for computing midsurface of a Stapler model. The household paper stapler has two parts, top and bottom. The bottom part is chosen for benchmarking as its midsurface computed from a commercial CAE system had errors. The intent of this exercise, is to compare it with the output generated by **MidAS**.

8.3.1 Input CAD Model

Stapler CAD model, which is input to **MidAS** is modeled using Sheet Metal modeling environment of Autodesk Inventor. Figure 8.23 shows the model and Figure 8.24 shows the corresponding feature tree.

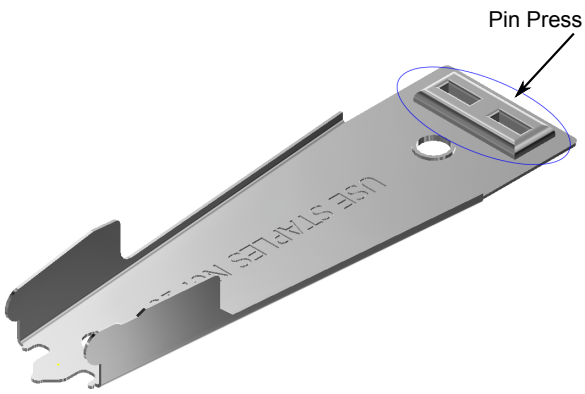


Figure 8.23: Input Sheet Metal CAD Model

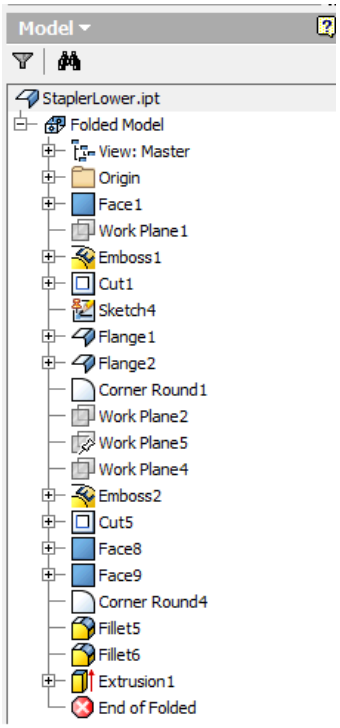


Figure 8.24: Feature Tree

Sheet metal features such as Face (Wall), Flange, Hole, Emboss, etc. have been used. Dependencies amongst features was minimized by not referencing faces or edges of previously modeled features, but by using reference geometries such as planes, axes, etc.

Following subsections show progress of this model through various modules of **MidAS**.

8.3.2 CAD Model Defeaturing

Defeaturing removes irrelevant features to compute the "gross shape". It also caches tool-bodies of relevant negative features to be used for piercing after midsurface computation.

As per defeaturing based on Sheet Metal feature taxonomy rules, elaborated in Section 4.4.1, secondary features chosen for removal, as seen in Figure 8.25 and 8.26, are:

- CornerRound1, CornerRound5
- Fillet5, Fillet6
- Cut1, Cut5
- Emboss1, Emboss2

Figure 8.25 also shows the identification of dormant features, two holes. As elaborated in Section 4.6, relevant negative features, such as Cutouts in the example shown

(features Cut1 and Cut5), are removed after storing their tool-bodies. These Dormant Features' tool-bodies, represented by Extrusion2 and Extrusion3, are shown in Figure 8.25.

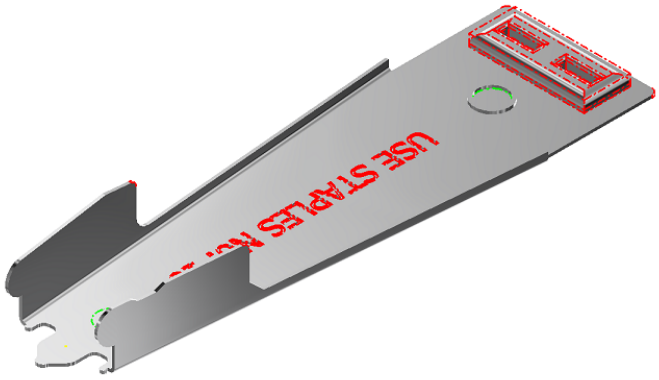


Figure 8.25: Features Selected for Removal

The ''Pin Press" site, as shown in Figure 8.25 presents peculiar case. This portion is not thin, is not part of oeverll shape, so should be removed. Only due to remnant feature approach this portion was detected and removed. Such removal is not seen in any reviewed approaches.

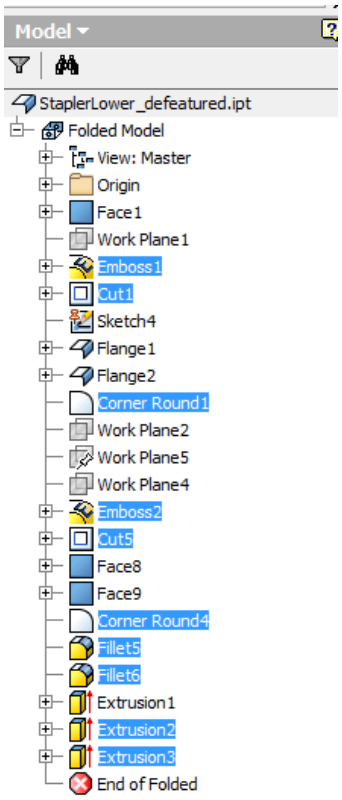


Figure 8.26: Feature Tree

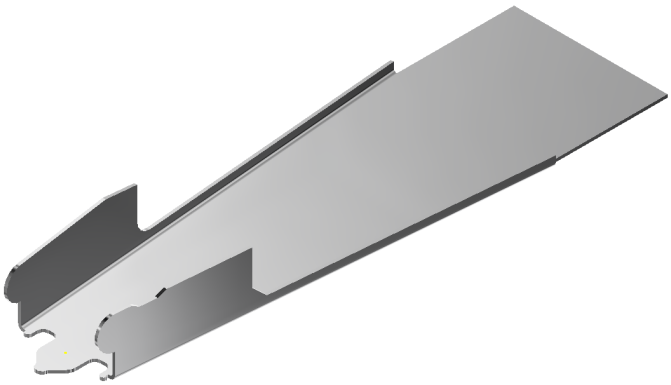


Figure 8.27: Defeatured Model

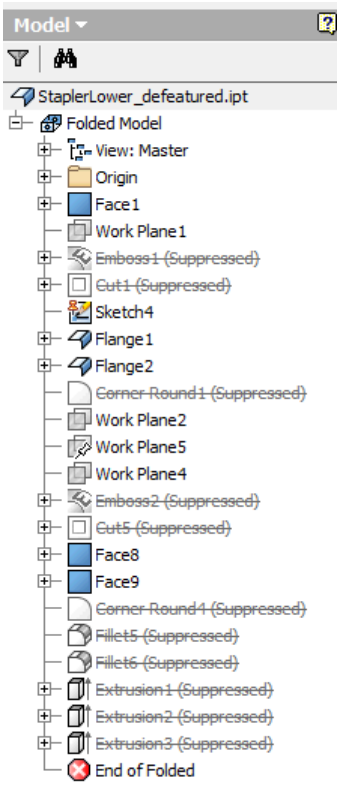


Figure 8.28: Feature Tree

Based on Eqn. 4.1. Defeaturing Effectiveness is computed as follows:
$pR = (1 - \frac{62}{296}) \times 100 = 79.05\%$

The output gross shape is shown in Figure 8.27. The overall shape and the design intent is well preserved in the gross shape.

8.3.3 CAD Model Generalization

The purpose of this module is to transform remaining sheet metal features into $\mathcal{ABLE}$ features. Figure 8.29b shows feature tree of the input defeatured model as well as the transformed $\mathcal{ABLE}$ features' tree.

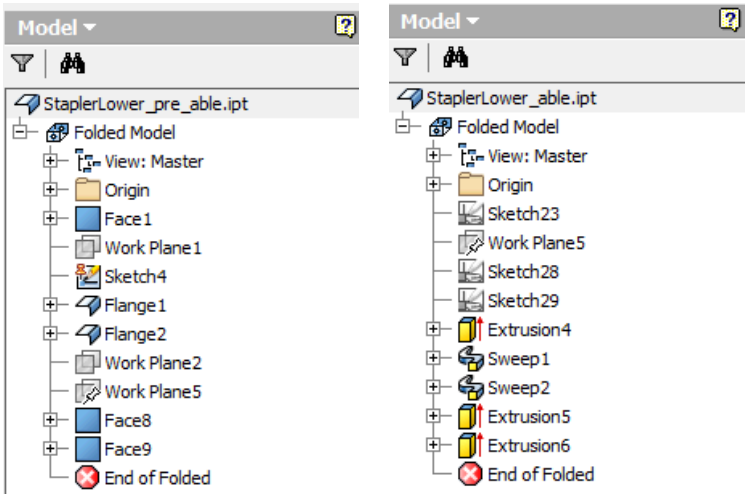
Table 8.3 shows how all Sheet Metal features are generalized to their corresponding $\mathcal{ABLE}$ such as Extrude, Sweep, etc. as per approach detailed in Section 5.4.6.

Table 8.3: Case II: CAD to $\mathcal{ABLE}$ Feature Mapping

CAD Feature	$\mathcal{ABLE}$ Feature
Face1	Extrusion4
Flange1	Sweep1
Flange2	Sweep2
Face8	Extrusion5
Face9	Extrusion6

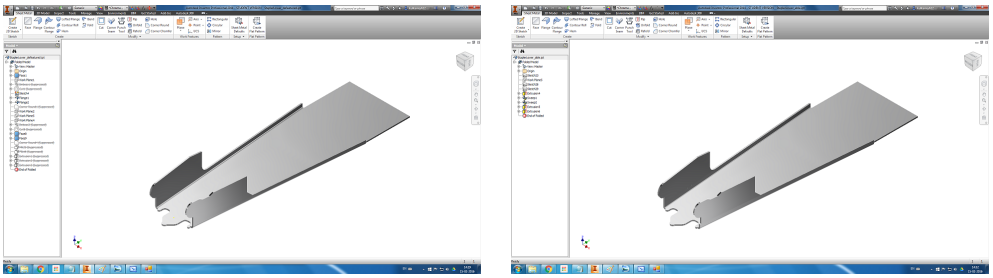
Figure 8.30a shows the input model having sheet metal features such as FACE, Flange, etc. Algorithms for transforming each of these features are listed in Table 5.2.

Thus, after transforming all the sheet metal features in the input model to their



(a) CAD Feature Tree (b) $\mathcal{ABLE}$ Feature tree

Figure 8.29: Transformation of CAD Model to Generalized Features



(a) Input Defeatured Model (b) Transformed to $\mathcal{ABLE}$ Model

Figure 8.30: Transformation of CAD Model to $\mathcal{ABLE}$ Model

$\mathcal{ABLE}$ features, the ouput $\mathcal{ABLE}$ model retains the overall shape and design intent.

8.3.4 CAD Model Decomposition

In the present research work the $\mathcal{ABLE}$ model is appropriately decomposed manually to create solid and interface cells. As elaborated in Section 6.3.2 the decomposition is done in two steps viz. Feature partitioning and convex partitioning. In feature partitioning "Unite" booleans are set to "New Body" type thereby separating the tool bodies of the features. Even after separating the feature tool bodies, there could be volumetric overlaps as seen in Figure 6.9b. In such cases, at this step, overlapping feature volumes are split at the concave edges, by convex partitioning. Figure 8.31 shows the decomposed model.

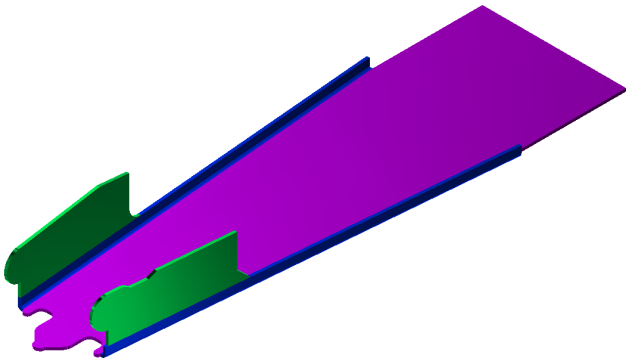


Figure 8.31: Cellular Decomposed Model

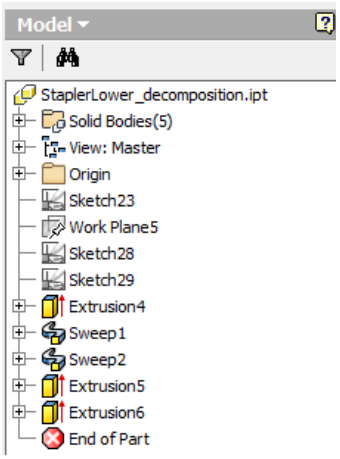


Figure 8.32: Feature Tree

Figure 8.31 decomposition of $\mathcal{ABLE}$ CAD model into cells. Figure 8.32 shows number of Solid Bodies as '5' denoting the number of cells. The cells are classified into midsurface patch generating cells and midsurface patch joining cells.

8.3.5 Midsurface Generation

After decomposition the model is in the form of collection of cells, with $\mathcal{ABLE}$ owner features. So, midsurface computation problem reduces to the problem of computing midsurface for individual cells.

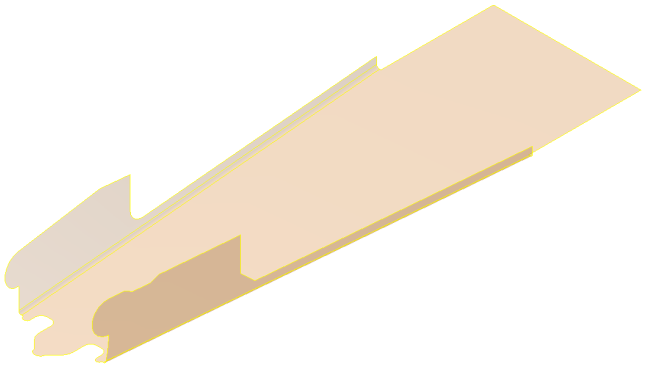


Figure 8.33: Output Connected Midsurface

Each cell, being a $\mathcal{ABLE}$ feature owned cell, the problem reduces to finding midsurface of a $\mathcal{ABLE}$ feature. Thus the complex problem gets reduced to a set of more manageable and deterministic sub-problems. After all cells are processed, the result is a well connected midsurface as shown in Figure 8.33.

8.3.6 Dormant Feature Re-application

Midsurface generated at this stage is not final yet. As elaborated in Section 4.4.1, during defeaturing, negative features are removed even though they are relevant. The

intent is to simplify and to make midsurface computation simpler. These removed features, called Dormant features, are reinstated back on the midsurface here.

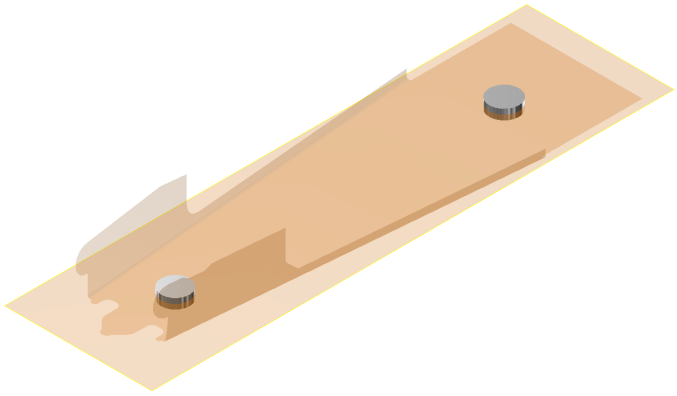


Figure 8.34: Dormant Features Re-application

Cached dormant feature tool bodies are pierced into the midsurface to restore the temporarily-suppressed negative features. Figure 8.34 shows the tool-bodies being re-applied on the midsurface, whereas Figure 8.35 shows the output final midsurface.

8.3.7 Final Output

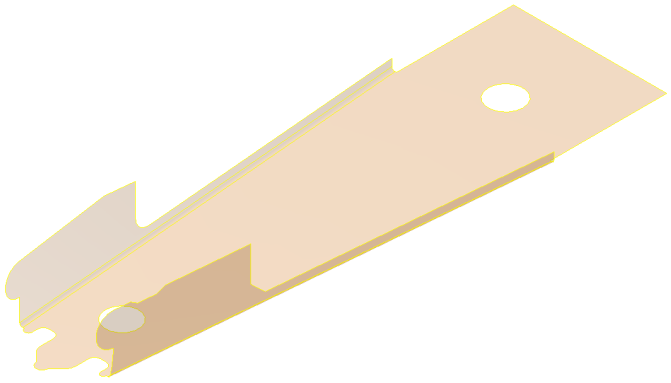


Figure 8.35: Final Midsurface

Figure 8.35 shows the final output midsurface. Compared to the commercial CAE system's output, **MidAS** has generated a better midsurface, without any extra patches. Figure 8.36 shows midsurface computed by a commercial CAE system. It shows two extra patches at the site of pin-press. These patches should not have been there.

Analysis of these failures suggests that at the site of pin-press, defeaturing did not happen properly and thus those midsurface patches got created at the irrelevant features, leaving extra patches in the output midsurface. Currently these patches have to be removed manually. Intent of this case study is to observe **MidAS**'s performance compared to the output shown in Figure 8.36.

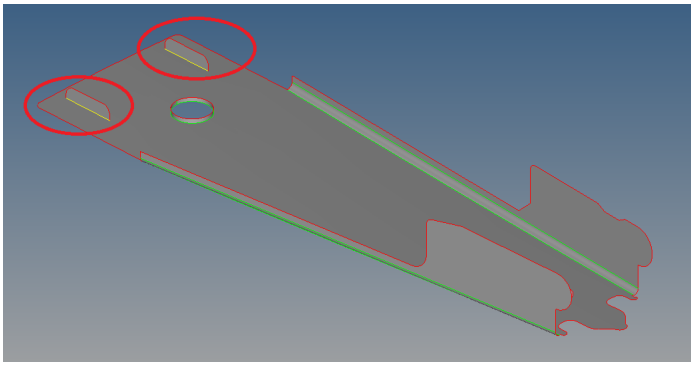


Figure 8.36: Midsurface Computed by Commercial CAE System

8.3.8 CAE Analysis

Finally the output midsurface is sent to CAE system, in which shell elements are applied on it. Figure 8.37 shows meshing, load applied as well as the boundary conditions specified along with the CAE analysis results. Figure 8.37 shows output of the analysis.

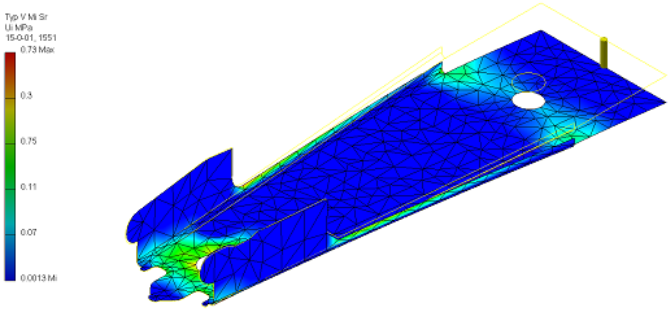


Figure 8.37: Result of the CAE Analysis on Computed Midsurface

Following are salient conclusions with regards to the test case elaborated above:

- Defeaturing module reduced complexity of the input model substantially (about 79% reduction in number of faces) and still retained the gross shape.
- Remnant feature's approach rightly selected the "Pin Press" site which made of **MidAS** better than the commercial system.
- Compared to the output by a commercial CAD-CAE system, the midsurface computed by **MidAS** clearly has a better output.

8.4 Case Study III

This case study tests **MidAS** for computing midsurface of a standing bracket part model, seen on GrabCAD® [140]. Figure 8.38 shows a real-life sheet metal part called Standing Bracket.

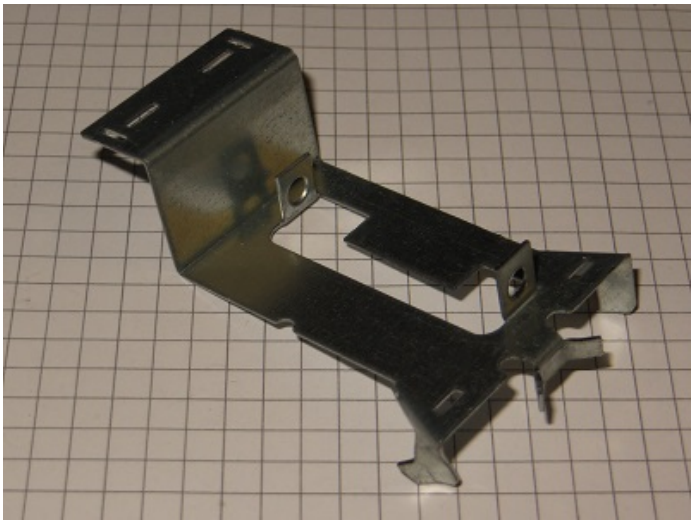


Figure 8.38: Standing Bracket Part

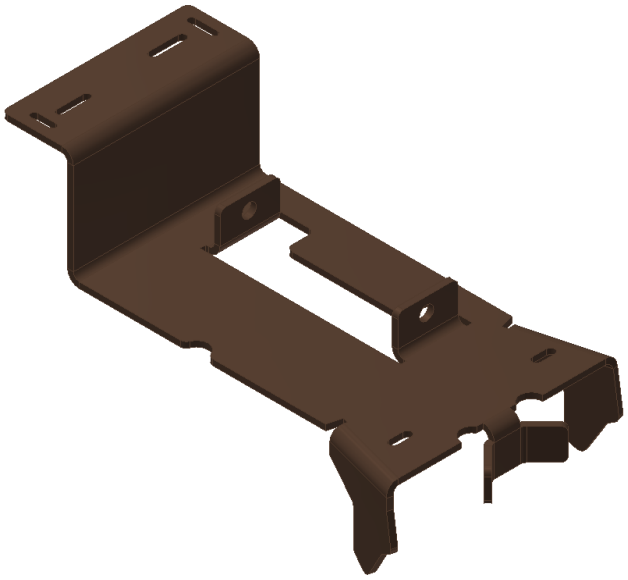


Figure 8.39: Input Model

Figure 8.39 shows CAD model built using Autodesk Inventor and Figure 8.40 shows its feature tree. It has sheet metal features like Face (Wall), Flange, Bend, Cutouts, etc.

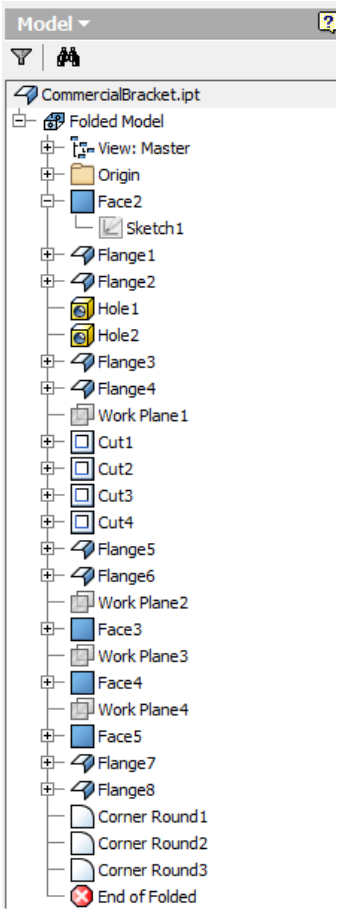


Figure 8.40: Feature Tree

As per defeaturing based on Sheet Metal feature taxonomy rules, elaborated in Section 4.4.1, secondary features chosen for removal, as seen in Figure 8.41 and 8.42, are:

- Cut1, Cut2, Cut3, Cut4. Peculiarity of this model is the cutout in the middle.
- CornerRound1, CornerRound2, CornerRound3

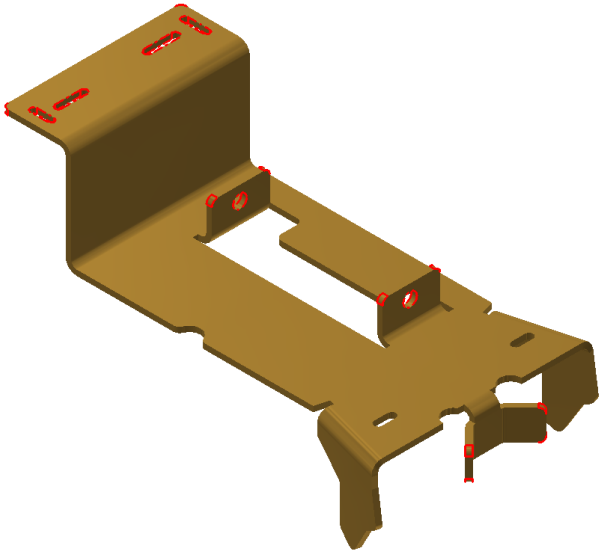


Figure 8.41: Input Model

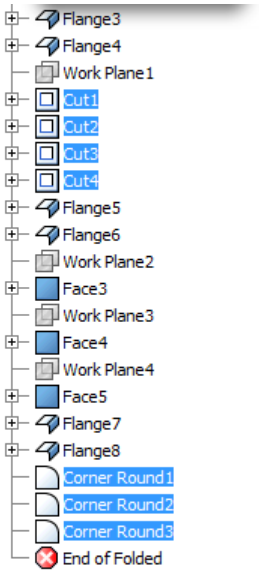


Figure 8.42: Feature Tree

As the middle cutout was modeled as a hole the sketch of Face(Wall) feature, it did not get the treatment of dormant feature.

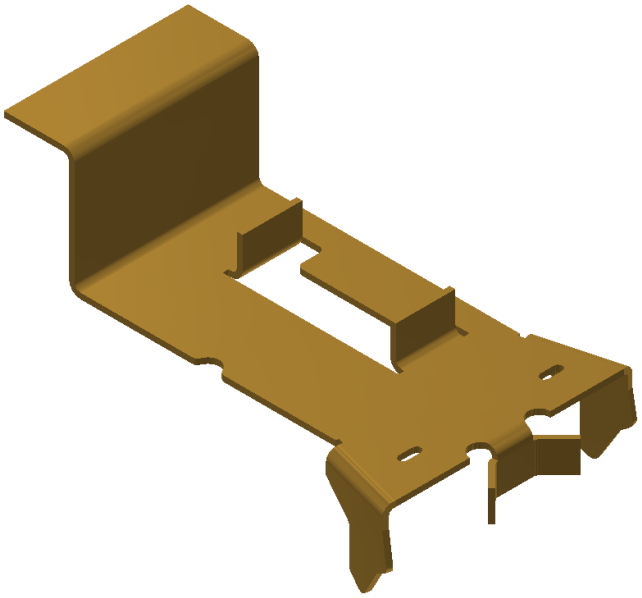


Figure 8.43: Input Model

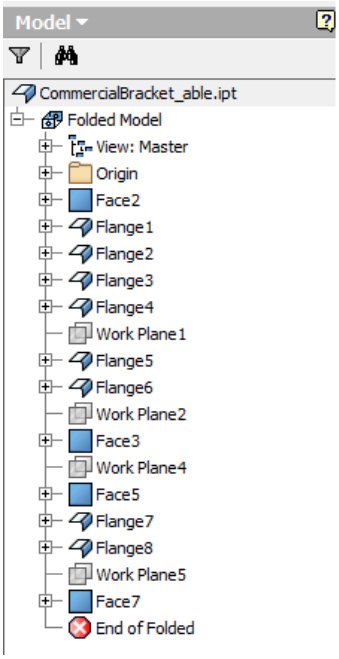


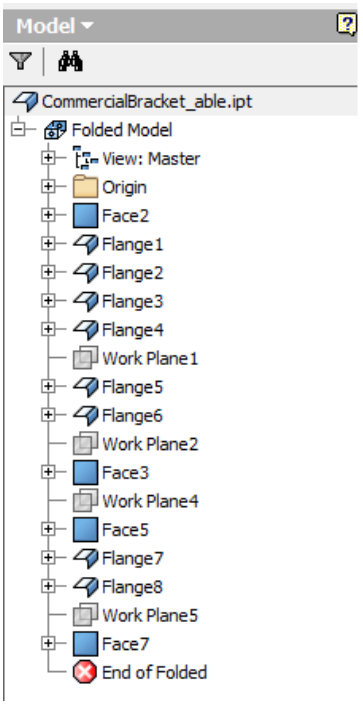
Figure 8.44: Feature Tree

Based on Eqn. 4.1. Defeaturing Effectiveness is computed as follows:

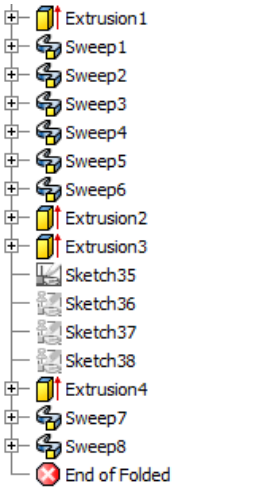
$pR = (1 - \frac{143}{171}) \times 100 = 16.03\%$

The output gross shape is shown in Figure 8.43 and corresponding tree in Figure 8.44. The overall shape and the design intent is well preserved in the gross shape.

Figure 8.45a shows the input CAD feature tree whereas Figure 8.45b shows the transformed feature tree.



(a) CAD Feature Tree



(b) $\mathcal{ABLE}$ Feature tree

Figure 8.45: Transformation of CAD Model to Generalized Features

Table 8.4 shows how all sheet metal features are generalized to their corresponding $\mathcal{ABLE}$ such as Extrude, Sweep, etc. as per approach detailed in Section 5.4.6. Algorithms for transforming each of these features are listed in Table 5.2.

Table 8.4: Case III: CAD to $\mathcal{ABLE}$ Feature Mapping

CAD Feature	$\mathcal{ABLE}$ Feature
Face2	Extrusion1
Flange1	Sweep1
Flange2	Sweep2
Flange3	Sweep3
Flange4	Sweep4
Flange5	Sweep5
Flange6	Sweep6
Face3	Extrusion2
Face5	Extrusion3
Flange7	Extrusion4
Flange8	Sweep7
Face7	Sweep85

Figure 8.46 shows transformation of $\mathcal{ABLE}$ CAD model into cells. A cellular graph is populated and the cells are classified into midsurface patch generating cells and midsurface patch joining cells.

After all cells are processed the result is a well connected midsurface as shown in Figure 8.47.

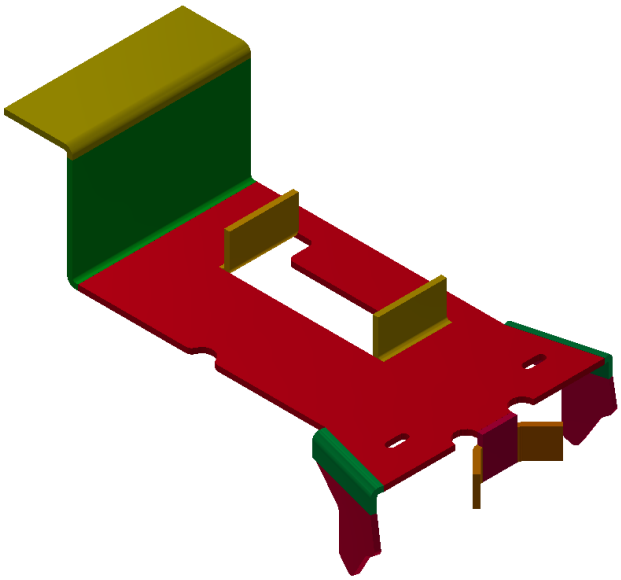


Figure 8.46: Decomposition of $\mathcal{ABLE}$ CAD Model into Cells

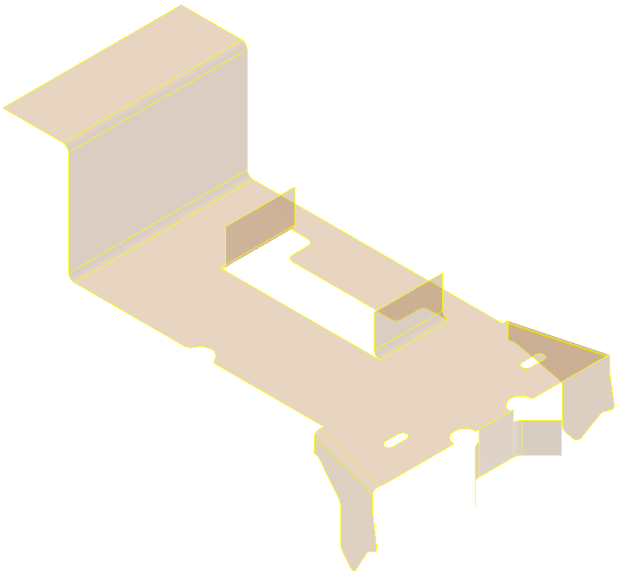


Figure 8.47: Output Connected Midsurface

This case study demonstrated full work-flow starting from a real-life part, up-to CAE analysis of the midsurface generated by **MidAS**.

8.5 Case Study IV

This case study tests **MidAS** for computing midsurface of a rectangular bracket part model, seen at the local sheet metal press shop. Figure 8.48 shows top and the side view of the part.

(a) Part-Top View (b) Part-Side View

Figure 8.48: Rectangular Bracket Part from a Small Scale Industry

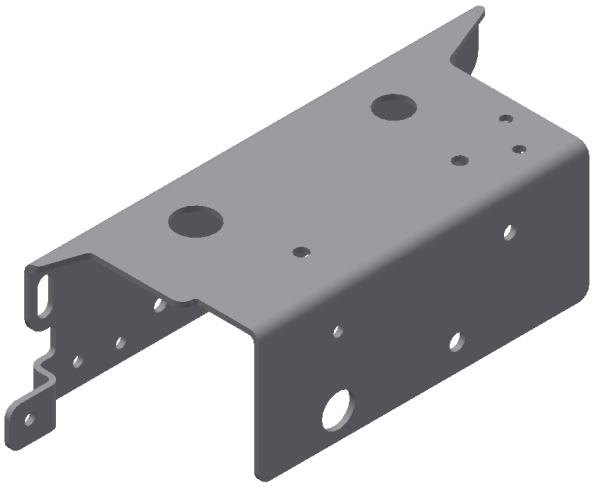


Figure 8.49: Input Model

Figure 8.49 shows CAD model built using Autodesk Inventor and Figure 8.50 shows corresponding tree. It has sheet metal features like Face (Wall), Flange, Bend, Hole, etc.

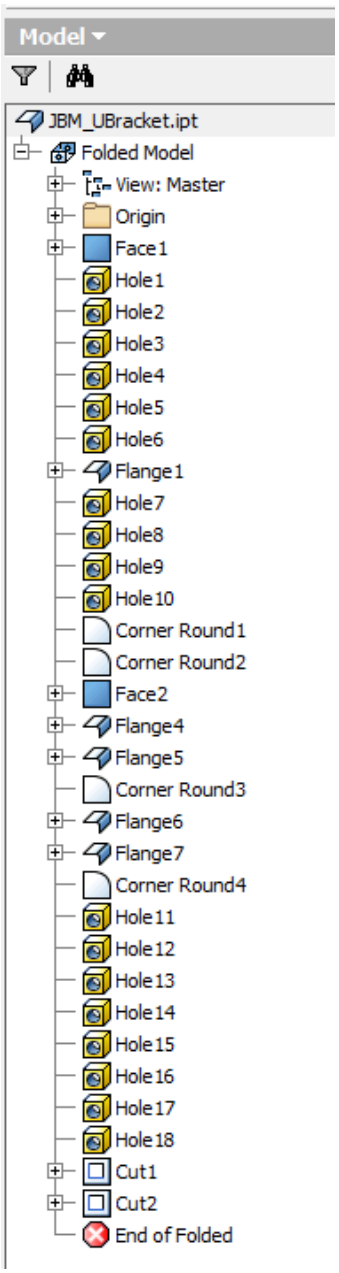


Figure 8.50: Feature Tree

As per defeaturing based on Sheet Metal feature taxonomy rules, elaborated in Section 4.4.1, secondary features chosen for removal, as seen in Figure 8.51 and 8.52, are:

- Hole1 to Hole18
- CornerRound1 to CornerRound4
- Cut1, Cut2

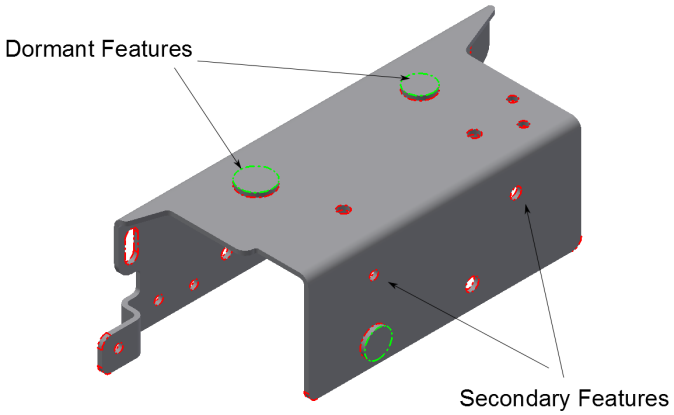


Figure 8.51: Input Model

Figure 8.51 shows features selected for Defeaturing. Small holes, slots, corner rounds, etc. are selected for removal. Dormant features identified are also shown in Figure 8.51. They are represented by Extrusion1, Extrusion2 and Extrusion3.

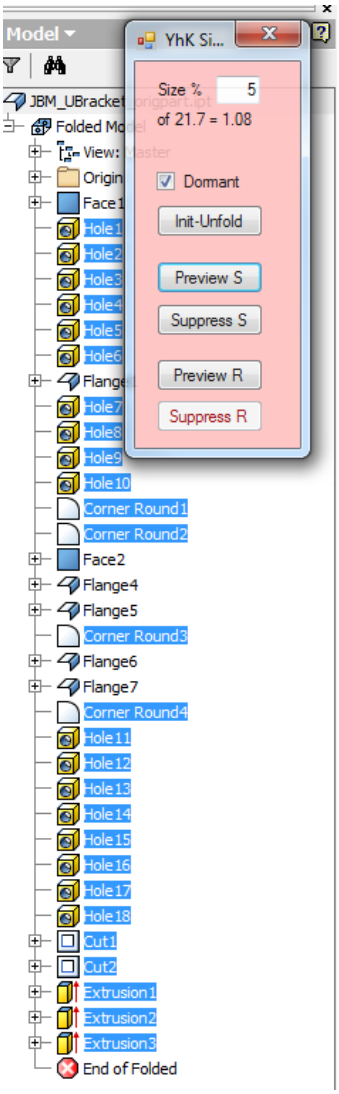


Figure 8.52: Feature Tree

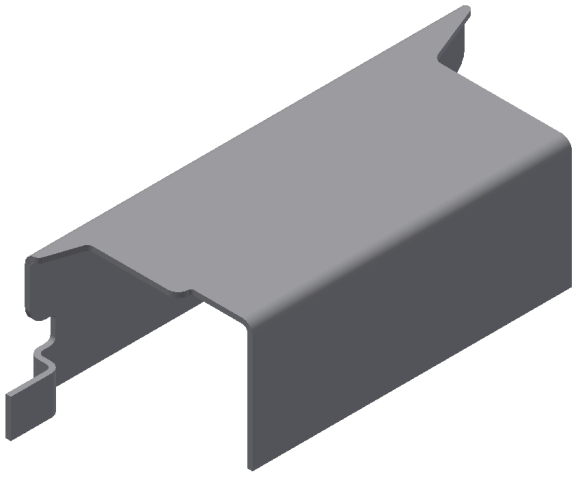


Figure 8.53: Input Model

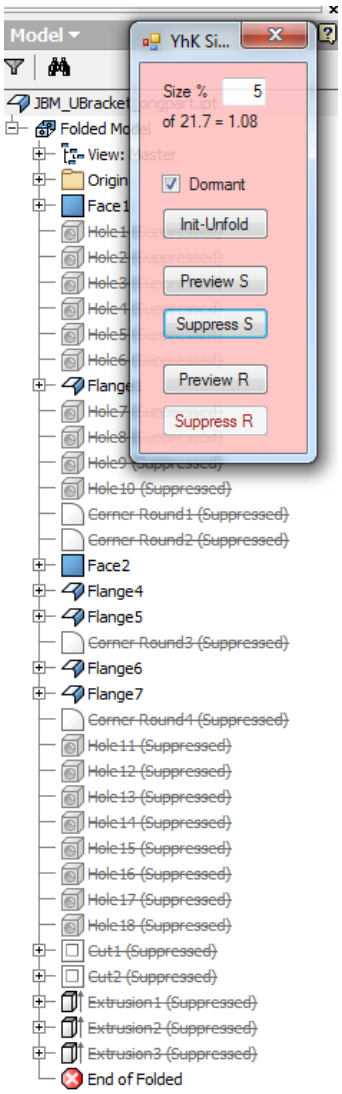


Figure 8.54: Feature Tree

The output gross shape is shown in Figure 8.53 and its feature tree is shown in Figure 8.54. The overall shape and the design intent is well preserved in the gross shape.

Based on Eqn. 4.1. Defeaturing Effectiveness is computed as follows:

$pR = (1 - \frac{77}{109}) \times 100 = 29.35\%$

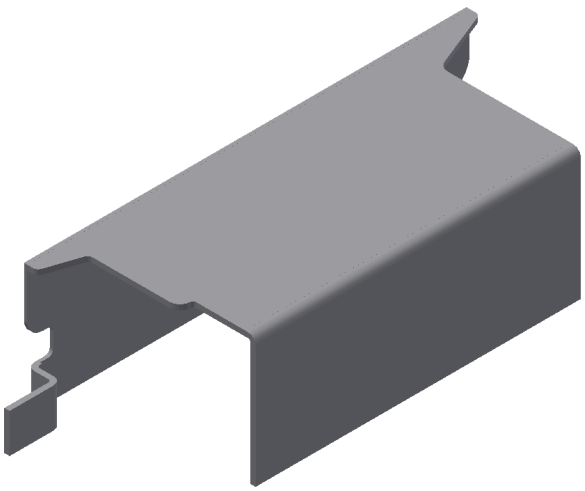


Figure 8.55: Input Model

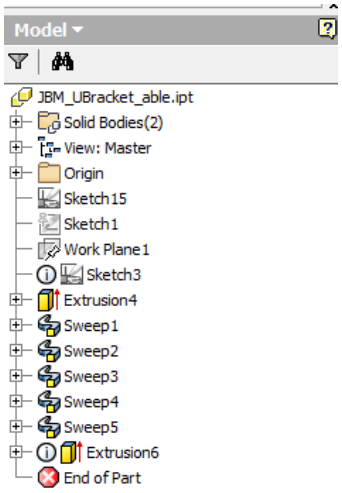


Figure 8.56: Feature Tree

Figure 8.55 shows transformation of gross shape features into generalized $\mathcal{ABLE}$ features tree (Figure 8.56). This output model is known as $\mathcal{ABLE}$ CAD model.

Figure 8.57 shows transformation of $\mathcal{ABLE}$ CAD model into cells. The cells are classified into midsurface patch generating cells and midsurface patch joining cells.

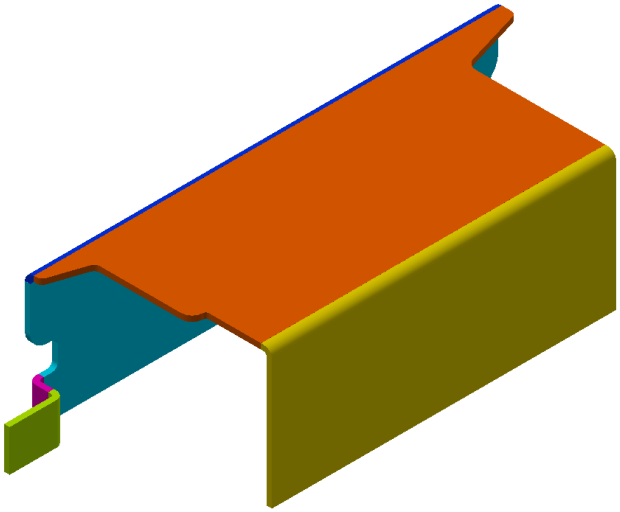


Figure 8.57: Decomposition of $\mathcal{ABLE}$ CAD Model into Cells

After all cells are processed the result is a well connected midsurface as shown in Figure 8.58.

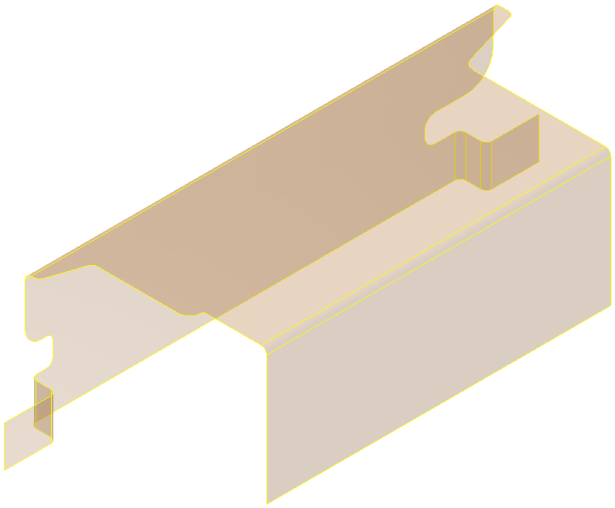


Figure 8.58: Output Conncted Midsurface

Midsurface generated at this stage is not final yet. As elaborated in Section 4.4.1, during defeaturing, negative features were removed even though they were relevant. The intent was to simplify and to make midsurface computation simpler. These removed features, called Dormant features, are reinstated back on the midsurface here. Cached dormant feature tool bodies are pierced into the midsurface to restore the temporarily-suppressed negative features. Figure 8.59 shows the tool-bodies being re-applied on the midsurface, whereas Figure 8.60 shows the output final midsurface.

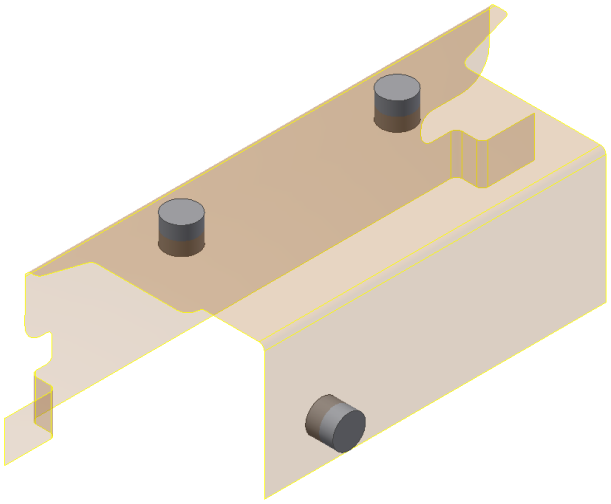


Figure 8.59: Dormant Feature Re-application

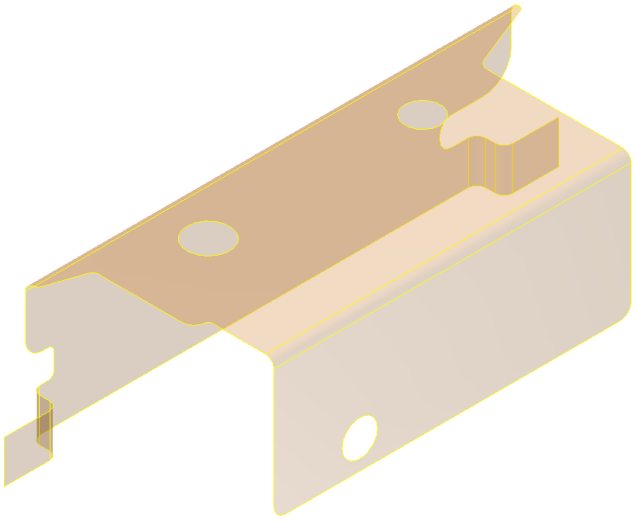


Figure 8.60: Final Midsurface Computed by **MidAS**

This case study demonstrated full work-flow starting from a real-life part, up-to final midsurface generated by **MidAS**.

8.6 Conclusions

The case studies presented in this chapter demonstrate working of the proposed system, called **MidAS** and its constituent modules such as defeaturing, generalization, decomposition, midsurface generation and dormant re-application. Results at each stage are analyzed and peculiarities of specific test cases are highlighted. Benchmarking with output generated by commercial systems is also presented for comparison. Finally CAE analysis of generated midsurface is shown to establish usefulness of the output.

The results obtained by **MidAS** are found to be deterministic and superior compared to those available in the commercial CAD-CAE systems.

Chapter 9

Conclusions

CAD models of thin-walled parts such as sheet metal or plastics are often reduced dimensionally to their corresponding midsurfaces for quicker and fairly accurate results of CAE analysis. Generation of the midsurface is still a time-consuming and mostly, a manual task due to lack of robust and automated techniques. Midsurface failures manifest in the form of gaps, overlaps, not-lying-halfway, etc., which can take hours or even days to correct. Thus, the need of a robust method of computing a well-connected midsurface has been felt by the CAE community. The present research work has addressed this need, specifically for the parts in sheet metal domain.

In the present research approach, input sheet metal feature-based CAD model is simplified first by defeaturing and making relevant negative features dormant. The defeatured model is further simplified by transforming the remaining sheet metal features into generalized features. Thus, the midsurface algorithm needs to handle only a limited set of generalized features. The generalized feature model is simplified one more level by cellular decomposition. The cells are classified and delegated with the tasks of creating midsurface patches and joining them, to form a well-connected midsurface. The quality of output midsurface is assessed by the newly developed topological validation method. The approach is implemented as a software system, called **MidAS**, using Autodesk Inventor API's. The algorithms for various modules are developed in VB and C#.Net programming languages. **MidAS** has been extensively tested for generating midsurface for sheet metal feature based CAD model, both from literature and industrial case studies. The output of **MidAS** was found to provide well-connected midsurfaces, in a more deterministic manner with minimum failures.

Major conclusions based on the present research work are listed below:

1. Most of the existing approaches of computing midsurface work on the Brep CAD models, which could be large and complex in case of real-life parts. Hard-coded heuristic rules are used and algorithms are developed on a case-by-case basis, resulting in substantial failures. The research presented here has resolved these problems by leveraging feature information, made available by the modern CAD

systems through APIs.

2. The present research has achieved resolution of two of the most critical issues observed in the widely used existing approach for midsurface generation, called Face Pairing. They are, face-pair detection and midsurface patches joining.

   MidAS addresses these by simplifying the input model in multiple ways and then devising a generic algorithm for computing a well connected midsurface.

3. Apart from addressing some critical issues specific to computation of midsurface, the present research has demonstrated a higher level paradigm for developing CAD algorithms. It leverages use of features, simplification, abstraction and decomposition as a general approach to be used in the development of CAD functionalities. This has been demonstrated in the present research at two places, one in the computation of midsurface and the second in the computation of midcurves. In the computation of midsurface, which is a 3D→2D, i.e. solid to surface transformation, input 3D solid model is simplified by removing irrelevant details. Abstraction by the way of generalization is then used to reduce a large variety of features to a limited set to work upon. Decomposition divides the model into manageable primitive 3D cells, which helps in devising more deterministic generic logic for computation. Same paradigm is used in case of computation of midcurves as well, which is a 2D→1D, i.e. profile to curve transformation. In simplification, free-from profile shape is faceted into a polygon. Then inner polygons are connected to the outer one to make one general polygonal shape. The polygon is then decomposed into 2D cells, where devising generic logic for computation is easier and deterministic. Thus, the new paradigm has been used effectively at two different dimensionalities and for two different computations. It also has potential to be used in other domains such as CAM and CAE.

Following section elaborates the contributions made by the present research.

9.1 Research Contributions

Contributions in the present research work are:

1. **Automatic defeaturing of sheet metal feature-based CAD model**: The defeaturing approach in the present research has made contributions to all three phases. In the first phase, newly developed sheet metal features taxonomy was introduced to decide removal of irrelevant features. In the second phase, newly devised approach for removing features based on size of the remnant portions was introduced. These two criteria combined together enabled removal of all the irrelevant features within the context, without compromising on the gross shape. In the

third phase, a new approach of temporarily removing features was introduced. Bulk negative features, called dormant features, were removed after storing their tool-bodies, to be used later for reapplication on the midsurface. The resultant model represented the gross shape, preserving the design intent of the input CAD model.

2. **Automatic transformation of sheet metal features to generalized features**: In order to make midsurface computation more generic, an automatic approach of transforming sheet metal features to generalized features was developed. A new generalized CAD model representation, called $\mathcal{ABLE}$ was introduced to generalize the input set of features to a limited set of generic features. The resultant model has same shape and size but is built only with generalized features of $\mathcal{ABLE}$ paradigm.

3. **Enhancement of existing Cellular Decomposition approach**: Existing approach of convex partitioning was leveraged for feature based cellular decomposition. The decomposed cell would now have an owner feature as well. A graph based cellular model was used for generalizing midsurface computation. Thus, whatever the size or complexity of the input CAD model may be, this step reduced it to a set of primitive shaped cells, which are easier to compute midsurface.

4. **Automatic computation of Midsurface**: An automatic, generic way of computing midsurface has been developed based on cells generated by decomposition. The cells were classified and delegated task of either computing midsurface patches or joining them. The patch generating cells, based on their owner feature's profile and guide, computed midsurface patches. Midsurface patch joining cells, based on a generic logic, joined the incident patches by creating appropriate surfaces between them.

5. **Automatic computation of Midcurve**: An automatic, generic way of computing midcurve has been developed. Input 2D profile was simplified by polygonization and then decomposed into sub-polygons. A generic logic was developed to compute midcurve segments and their joining, to form a well-connected midcurve.

6. **Topological Validation**: Introduced topological validation method for assessing quality of the output midsurface, to be used along with the existing geometric validation methods. Transformation equations were developed to predict topological entities of output midsurface, given the topological entities of the input model, and vice versa. The predicted entities were compared against the actual entities to detect any failures in the output midsurface.

9.2 Scope for Further Work

The present research work, **MidAS**, has addressed the critical issues of the existing approaches of computation of midsurface by effective use of a new paradigm. However, in order to broaden the scope, following enhancements can be incorporated:

1. More sheet metal features can be incorporated in the generalization module, which will increase **MidAS**'s applicability to a wider range of parts.

2. Development of **MidAS** using APIs of other CAD-CAE applications will further increase its applicability.

3. Extending the scope of **MidAS** to include domains such as injection molding, where better midsurface is highly desirable.

MidAS has presented an integrated approach to generation of well-connected midsurfaces of sheet metal CAD models using newly developed paradigm. It has thus enabled an important step towards the automation and integration of CAD-CAE activities for Product Development Process.

References

[1] Roland Stolt. *CAD-Model parsing for Automated Design and Design Evaluation*. PhD thesis, Jonkoping University, 2008. x, 2, 36, 67

[2] Trevor Robinson Christopher Tierney, Declan Nolan and Cecil Armstrong. Using mesh-geometry relationships to transfer analysis models between cae tools. *Proceedings of the 22nd International Meshing Roundtable*, 2013. x, 3

[3] Yoonhwan Woo. Abstraction of mid-surfaces from solid models of thin-walled parts: A divide-and-conquer approach. *Comput. Aided Des.*, 47:1–11, 2014. doi: 10.1016/j.cad.2013.08.010. x, 7, 8, 28, 29, 31, 123, 203

[4] Tony Abbey. The art of idealization in finite element analysis. Technical report, Desktop Engineering, 2013. x, xiv, 9, 11, 201

[5] M Hamdi, N Aifaoui, and A Benamara. Design and analysis integration model based on idealization of cad geometry. *Intl conf on advances in mech engineering and mechanics (ICAMEM)*, 2006. x, 9, 10, 14, 16, 30

[6] Dong-Pyoung Sheen, Tae-geun Son, Cheolho Ryu, Sang Hun Lee, and Kunwoo Lee. Dimension reduction of solid models by mid-surface generation. *International Journal of CAD/CAM*, 7(1):71–80, 2009. x, xii, 11, 29, 94

[7] Huawei Zhu, Yanli Shao, Yusheng Liu, and Jianjun Zhao. Automatic hierarchical mid-surface abstraction of thin-walled model based on rib decomposition. *Advances in Engineering Software*, 97:60–71, 2016. x, 15, 22, 24, 92, 93

[8] M Ramanathan and B Gurumoorthy. Generating the mid-surface of a solid using 2D MAT of its faces. *Computer Aided Design and Applications*, 1:665–674, 2004. doi: 10.1080/16864360.2004.10738312. x, 8, 18, 20, 23, 109

[9] M Ramanathan and Balan Gurumoorthy. Constructing medial axis transform of extruded and revolved 3d objects with free-form boundaries. *Computer-Aided Design*, 37(13):1370–1387, 2005. x, 18

[10] Dong-Pyoung Sheen, Tae geun Son, Dae-Kwang Myung, Cheolho Ryu, Sang Hun Lee, Kunwoo Lee, and Tae Jung Yeo. Solid deflation approach

to transform solid into mid-surface. *Proceedings of the TMCE*, 2010. doi: 10.1016/j.cad.2010.01.003. x, 20, 25, 29, 91

[11] Quadros, William Roshan, Steven James Owen, Mike Brewer, and Kenji Shimada. Finite element mesh sizing for surfaces using skeleton. In *Proceedings, 13th International Meshing Roundtable*, pages 389–400, September 2004. doi: 10.1007/s00366-004-0292-4. x, 21, 23

[12] Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A novel type of skeleton for polygons. 1996. x, 22, 23

[13] A. Fischer and K. K. Wang. A method for extracting and thickening a mid-surface of a 3d thin object represented in nurbs. *Journal of manufacturing science and engineering, Transactions of ASME*, 119:706–712, 1997. doi: 10.1115/1.2836813. x, 22, 24

[14] Flavien Boussuge. *Idealization of CAD assemblies for FE structural analyses*. PhD thesis, University of Grenoble, 2006. x, xi, 25, 31, 92

[15] Roland Stolt. Reusing cad models for die-casting products for fea. *Proceedings of 2nd NAFEMS seminar: prediction and modelling of failure using FEA*, May 2006. x, 5, 26, 27, 96

[16] F. Boussuge, J.-C. Lon, S. Hahmann, and L. Fine. Idealized models for fea derived from generative modeling processes based on extrusion primitives. *Proceedings of the 22nd international meshing roundtable*, pages 127–145, 2014. doi: 10.1007/978-3-319-02335-9-8. x, 28, 31, 69

[17] Masaru Kageura and Shi Yokohoma. Apparatus and method for generating analysis model, 2009. x, 28, 29, 30

[18] Sankara Hari Gopalakrishnan and Krishnan Suresh. A formal theory for estimating defeaturing-induced engineering analysis errors. *Computer-Aided Design*, 39(1):60–68, January 2007. doi: 10.1016/j.cad.2006.09.006. x, 45, 46

[19] Autodesk. Autodesk inventor 2014 help. Technical report, Autodesk, 2014. URL `http://help.autodesk.com/view/INVNTOR/2014/ENU/`. (Last accessed Oct 2015). x, 50, 67, 83

[20] Frank Hoisl and Kristina Shea. Exploring the integration of shape grammar and open source cad systems. In *International Conference on Engineering Design, ICED*, 2009. xi, 70

[21] Yan Wang Sean Tessier. Ontology-based feature mapping and verification between cad systems. *Advanced Engineering Informatics*, 27:76–92, 2013. xi, 71

[22] Yoonhwan Woo. Fast cell-based decomposition and applications to solid modeling. *Computer-aided Design*, 35:969–977, 2003. doi: 10.1016/S0010-4485(02)00144-6. xii, 99, 101

[23] Rafael Bidarra, Willem J Neels, and Willem F Bronsvoort. Boundary evaluation for a cellular model. pages 255–265, 2003. xii, 100

[24] Eelco van den Berg. Web based collaborative modelling with spiff. *Delft Univ. of Techy*, pages 1–52, 2000. xii, 100

[25] CGAL. Convex partitioning, 2013. URL `http://doc.cgal.org/latest/Partition_2/index.html`. http://doc.cgal.org/latest/Partition_2/index.html (Last accessed June 2016). xii, 112

[26] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Comput. Geom. Theory Appl.*, 35(1):100–123, August 2006. ISSN 0925-7721. doi: 10.1016/j.comgeo.2005.10.005. xii, 110, 113

[27] Helen Lockett and Marin Guenov. Similarity measures for mid-surface quality evaluation. *Comput. Aided Des.*, 40(3):368–380, March 2008. ISSN 0010-4485. doi: 10.1016/j.cad.2007.11.008. xiii, 32, 33, 129, 130, 131, 132, 133, 134, 143

[28] Ron K C Cheng. *Autodesk Inventor Tutorial, SHEET METAL MODELING*. Autodesk. xiv, 200

[29] M. Sypkens Smit. *Efficient remeshing and analysis views for integration of design and analysis*. PhD thesis, TU Delft, 2011. xiv, 16, 30, 202, 203

[30] Rich Austreng. Rapid generation of finite element models from 3d cad. Technical report, Altair Hyperworks, 2007. xiv, 5, 202

[31] Marc Halpern. Industrial requirements and practices in finite element meshing: A survey of trends. In *6th International meshing roundtable*, pages 399–411,, 1997. 3, 5, 11

[32] P Elangovan. A study on solid and shell material model in stamping simulation. Technical report, Valeo HTC, 2012. 4

[33] Ming Li, Junzhe Zheng, and Ralph R Martin. Quantitative control of idealized analysis models of thin designs. *Computers & Structures*, 106:144–153, 2012. 5

[34] Andr Backes, Christian Glockner, and Ulrich Schwanecke. Automatic mid-surface generation as a basis for technical simulation. *ATZ Worldwide*, 116(2): 20–23, 2014. doi: 10.1007/s38311-014-0019-0. 5, 20, 23

[35] ISD Group. Hicad - from the concept to the finished sheet metal product. URL `https://www.isdgroup.com/en/fields_of_application/sheet_metal.html`. (Last accessed 20 June 2016). 5

[36] Mohsen Rezayat. Midsurface abstraction from 3D solid models: general theory, applications. *Computer-aided Design*, 28:905–915, 1996. doi: 10.1016/0010-4485(96)00018-8. 8, 19, 25, 29, 32, 91, 92, 129

[37] SolidWorks. Analyzing sheet metal parts and assys in cosmosworks. Technical report, Dassault Systems, 2006. 9, 202

[38] M. Hamdi, N. Aifaoui, and A. Benamara. Idealization of cad geometry using design and analysis integration features models. *International Conference of Engineering Design*, 2007. 9, 14, 16, 30

[39] Hamdi Mounir, Aifaoui Nizar, B Louhichi, and Benamara Abdelmajid. Cad/cae interoperability, an automatic generation of analysis model based on idealization of cad geometry. *19th Congress Francais de Mecanique*, 2009. 16, 30

[40] M. Hamdi, N. Aifaoui, B Louhichi, and A Benamara. Simplification of cad geometry using an hybrid method for efficient integration to fem model. *Proceedings of IDMME*, 2010. 16, 30

[41] Mounir Hamdi, Nizar Aifaoui, Borhen Louhichi, and Abdelmajid BenAmara. Idealization of CAD model for a simulation by a finite element method. *European Journal of Control*, 19:419–439, 2010. doi: 10.3166/ejcm.19.419-439. 9, 14, 16, 30

[42] Kunwoo Lee, Yong-gu; Lee. Geometric detail suppression by the Fourier transform. *Computer-aided Design*, 30:677–693, 1998. doi: 10.1016/S0010-4485(98)00022-0. 11

[43] Mohamed Belaziz, Abdelaziz Bouras, and Jean-Marc Brun. Morphological analysis for product design. *Computer-Aided Design*, 32(5-6):377–388, 2000. doi: 10.1016/S0010-4485(00)00019-1. 11, 14, 16

[44] Y-M Deng, YC Lam, Shu Beng Tor, and GA Britton. A cad-cae integrated injection molding design system. *Eng with Computers*, 18(1):80–92, 2002. 11

[45] Sang Hun Lee. Feature-based multiresolution modeling of solids. *ACM Trans on Graphics*, 24:1417–1441, 2005. doi: 10.1145/1095878.1095887. 15, 17, 59, 60

[46] Sang Hun Lee. Feature-based non-manifold modeling system to integrate design and analysis of injection molding products. *Journal of Mechanical Science and Technology*, 23:1331–1341, 2009. doi: 10.1007/s12206-009-0407-3. 11, 15, 17

[47] Atul Thakur, Ashis Gopal Banerjee, and Satyandra K. Gupta. A survey of CAD model simplification techniques for physics-based simulation applications. *Computer-aided Design*, 41:65–80, 2009. doi: 10.1016/j.cad.2008.11.009. 12

[48] Yeonuk Kang, ByungChul Kim, Duhwan Mun, and Soonhung Han. Method to simplify ship outfitting and offshore plant equipment three-dimensional (3-d) computer-aided design (cad) data for construction of an equipment catalog. *Journal of Marine Science and Technology*, 19(2):185–196, 2014. ISSN 0948-4280. doi: 10.1007/s00773-013-0239-9. 12, 13, 16

[49] Padmanabh Dabke, Vallury Prabhakar, and Sheri Sheppard. Using features to support finite element idealizations. *Computers in Engineering*, 1:183–183, 1994. 13, 15, 30, 201

[50] Sypkens Smit. Integration of design and analysis models. *Computer-aided Design and Applications*, 6, 2009. doi: 10.3722/cadaps.2009.795-808. 13, 16, 30

[51] A.L. Ames, J.J. Rivera, A.J. Webb, and D.M. Hensinger. Solid model design simplification. Sandia Report SAND97-3141, Sandia Natl Labs, 1997. 13, 16

[52] Hamdi Mounir, Aifaoui Nizar, and Benamara Abdelmajid. CAD model simplification using a removing details and merging faces technique for a FEM simulation. *Journal of Mechanical Science and Technology*, 26(11):3539–3548, 2012. doi: 10.1007/s12206-012-0869-6. 13, 14, 16, 30

[53] Brian Russ. Development of a cad model simplification framework for finite element analysis. Master's thesis, Univ of Maryland, 2012. 13, 54, 59, 60

[54] Florence Danglade, Jean-Philippe Pernot, and Philippe Vron. On the use of machine learning to defeature cad models for simulation. *Comp-Aided Des and Appl*, 11:358–368, 2013. doi: 10.1080/16864360.2013.863510. 13, 16

[55] Yoonhwan Woo. Automatic simplification of solid models for engineering analysis independent of modeling sequences. *Journal of Mech Sc and Tech*, 23: 1939–1948, 2009. doi: 10.1007/s12206-009-0509-y. 14, 17, 29, 99, 101

[56] PA Watterberg. Geometric simplification of analysis models. Technical report, Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US), 1999. 14, 16

[57] Nikhil Joshi and Debasish Dutta. Feature Simplification Techniques for Freeform Surface Models. *Journal of Computing and Information Science in Engineering*, 3:177–186, 2003. doi: 10.1115/1.1603307. 14, 16

[58] Huawei Zhu, Yanli Shao, Yusheng Liu, and Chunguang Li. Mid-surface abstraction for complex thin-wall models based on virtual decomposition. *International Journal of Computer Integrated Manufacturing*, 29:821–838, 2015. doi: 10.1080/0951192X.2015.1068455. 14, 22, 24, 31

[59] Sang Hun Lee. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. *Computer-aided Design*, 37:941–955, 2005. doi: 10.1016/j.cad.2004.09.021. 15, 17, 55

[60] Byung Chul Kim and Duhwan Mun. Feature-based simplification of boundary representation models using sequential iterative volume decomposition. 38:97–107, February 2014. ISSN 0097-8493 (print), 1873-7684 (electronic). doi: 10.1016j.cag.2013.10.031. 15, 17

[61] Jae Yeol Lee, Joo-Haeng Lee, Hyun Kim, and Kim Hyung-Sun. A cellular topology-based approach to generating progressive solid models from feature-centric models. *Computer-Aided Design*, 36:217–229, 2004. 16

[62] Mark Price, Clive Stops, and Geoffrey Butlin. A media object toolkit for meshing and other applications. Technical report, FEGS Ltd. Oakington Cambridge CB4 5AF England, 2009. 18

[63] Cecil G Armstrong, Desmond J Robinson, RM McKeag, TS Li, SJ Bridgett, RJ Donaghy, and CA McGleenan. Medials for meshing and more. In *4th Int. Meshing Roundtable*, pages 277–288, 1995. 18

[64] Harry Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn, editor, *Proc. models for the perception of speech and visual form*, pages 362–380, Cambridge, MA, November 1967. MIT Press. URL `http://pageperso.lif.univ-mrs.fr/~edouard.thiel/rech/1967-blum.pdf`. 19, 20, 21, 23

[65] Lakshman Prasad. Rectification of the chordal axis transform skeleton and criteria for shape decomposition. *Image and Vision Computing*, 25(10):1557–1571, 2007. 19, 21, 23

[66] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885, September 1992. ISSN 0162-8828. doi: 10.1109/34.161346. 19

[67] Trevor T Robinson, Cecil G Armstrong, G McSparron, A Quenardel, Hengan Ou, and RM McKeag. Automated mixed dimensional modelling for the finite element analysis of swept and revolved cad features. In *Proceedings of the 2006 ACM symposium on solid and physical modeling*, pages 117–128. ACM, 2006. doi: 10.1145/1128888.1128905. 20, 23, 67

[68] Trevor Robinson and Cecil Armstrong. Automated complex mixed-dimensional model creation. Technical report, Queen's Univ of Belfast, 2007. 20, 23, 27

[69] Felix Stanley. *Dimensional reduction and design optimization of gas turbine engine casings for tip clearance studies*. PhD thesis, University of Southampton, 2010. 20, 23, 203

[70] Dominique Attali, Jean-Daniel Boissonnat, and Herbert Edelsbrunner. Stability and computation of the medial axis: A state-of-the-art report. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, Springer-Verlag*, 2004. 20

[71] Roger C. Tam and Wolfgang Heidrich. Shape simplification based on the medial axis transform. *IEEE Visualization*, pages 672–686, 2003.

[72] J. W. Brandt and V. R. Algazi. Continuous skeleton computation by voronoi diagram. *CVGIP: Image Understanding*, 55(3):329–338, 1992.

[73] J. W. Brandt. Describing a solid with the three-dimensional skeleton. *Proceedings of the SPIE Curves and Surfaces in Computer Vision and Graphics III*, 1830: 258–269, 1992.

[74] R Ogniewicz. Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto voronoi skeletons. *Tech. Rep*, 95(4), 1995.

[75] Roger C. Tam and Heidrich Wolfgang. Feature-preserving medial axis noise removal. *In ECCV 02: Proceedings of the 7th European Conference on Computer Vision-Part II*, pages 672–686, 2002.

[76] J. M. Reddy and G. M. Turkiyyah. Computation of 3d skeletons using a generalized delaunay triangulation technique. *Computer-Aided Design*, 27(9):677–694, 1995.

[77] G.M. Turkiyyah, D.W. Storti, M. Ganter, H. Chen, and M. Vimawala. An accelerated triangulation method for computing the skeletons of free-form solid models. 29(1):5–19, January 1997.

[78] Ju-Hsien Kao. *Process Planning For Additive/Subtractive Solid Freeform Fabrication Using Medial Axis Transform*. PhD thesis, Stanford Univ, 1999. 20

[79] William Roshan Quadros, Kenji Shimada, and Steven James Owen. Skeleton-based computational method for the generation of a 3D finite element mesh sizing function. *Eng with Comp*, 20(3):249–264, September 2004. 21, 23

[80] William Roshan Quadros. An approach for extracting non-manifold mid-surfaces of thin-wall solids using chordal axis transform. *Eng. with Comput.*, 24(3):305–319, 2008. ISSN 0177-0667. doi: 10.1007/s00366-008-0094-1. 21, 23

[81] W. R. Quadros, K. Ramaswami, F.B. Prinz, and B.Gurumoorthy. LayTracks: a new approach to automated quadrilateral mesh generation using MAT. In *Procs, 9th Intl Meshing Roundtable*, pages 239–250, 2000. 21, 23

[82] U. Montanari. Continuous skeletons from digitized images. *Journal of the Association for Computing Machinery*, 16(4):534–549, 1969. 21, 23

[83] H. N. Gursoy. *Shape Interrogation by Medial Axis Transform for Automated Analysis.* PhD thesis, Massachusetts Institute of Technology, 1989. 21, 23

[84] Oswin Aichholzer, Wolfgang Aigner, Franz Aurenhammer, Thomas Hackl, Bert Jüttler, and Margot Rabl. Medial axis computation for planar free–form shapes. *Computer-Aided Design*, 41(5):339–349, 2009. 22, 23

[85] A Fischer, A Smolin, and G Elber. Mid-surface of profile-based freeforms for mold design. *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, 121:202–207, 1999. doi: 10.1115/1.2831206. 22, 30, 121, 123

[86] SC Kim, KW Lee, TS Hong, MC Kim, MK Jung, and YJ Song. An integrated approach to realize multi-resolution of b-rep model. *Transactions of the Society of CAD/CAM Engineers*, 11(4):289–302, 2006. 26, 29, 59, 60

[87] Roland Stolt. A cad-integrated system for automated idealization of cad-models for finite element analysis. *Proceedings of DETC2005: ASME International Design Engineering Technical Conferences and Computers and Information in Engineering conferences*, September 2005. 27, 30, 200

[88] C. S. Chong, A. Senthil Kumar, and K. H. Lee. Automatic solid decomposition and reduction for non-manifold geometric model generation. *Computer-aided Design*, 36:1357–1369, 2004. doi: 10.1016/j.cad.2004.02.005. 28, 30

[89] Weijuan Cao, Haipang Wu, Yuqin Jiang, Yusheng Liu, and Shuming Gao. Representation and automated generation of analysis feature model for finite element analysis. *Proceedings of the ASME 2009 international design engineering technical conferences & computers and information in engineering conference*, 5: 605–614, Aug 2009. doi: 10.1115/DETC2009-86322. 28, 31, 67, 69

[90] Weijuan Cao, Xiaoshen Chen, and Shuming Gao. An approach to automated conversion from design feature model to ananlysis feature model. *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 5:655–665, 2011. doi: 10.1115/DETC2011-47555. 28, 31

[91] Flavien Boussuge, Jean-Claude Léon, Stéfanie Hahmann, and Lionel Fine. Extraction of generative processes from b-rep shapes and application to idealization transformations. *Comput. Aided Des.*, 46:79–89, January 2014. ISSN 0010-4485. doi: 10.1016/j.cad.2013.08.020. 28, 31

[92] Yoonhwan Woo and Sang Hun Lee. Volumetric modification of solid cad models independent of design features. *Advances in Engineering Software*, 37:826–835, 2006. 29, 99, 101

[93] Dong-Pyoung Sheen, Tae-geun Son, Dae-Kwang Myung, Cheolho Ryu, Sang Hun Lee, Kunwoo Lee, and Tae Jung Yeo. Transformation of a thin-walled solid model into a surface model via solid deflation. *Comput. Aided Des.*, 42(8): 720–730, August 2010. ISSN 0010-4485. doi: 10.1016/j.cad.2010.01.003. 29

[94] Aimin Ji, Kun Zhu, Xinlei Huang, and Xu Yin. Integration design and analysis of excavator boom based on cad/cae. *Journal of Theoretical and Applied Information Technology*, pages 138–143, 2013. 30

[95] Roland Stolt. A sectioning method for constructing the mid-surface of thin walled die-cast and injection moulded parts. Technical report, Jonkoping University, 2006. URL `http://www.diva-portal.org/smash/get/diva2:4318/FULLTEXT01.pdf`. 30

[96] F Boussuge, J-C Lion, S. Hahmann, and L. Fine. An analysis of dmu transformation requirements for structural assembly simulations. In *8th International Conference on Engineering Computational Technology (ECT'12)*, 2012. 31

[97] Autodesk. Techniques for electronic analyses, 2014. URL `http://knowledge.autodesk.com/support/simulation-cfd/learn-explore/`. Last accessed 28 July 2014. 48

[98] Ravi Kumar Gupta and Balan Gurumoorthy. Unified taxonomy for reference ontology of shape features in product model. pages 295–307, 2013. 49

[99] R. K. Gupta and B Gurumoorthy. Classification, representation, and automatic extraction of deformation features in sheet metal parts. *Computer-Aided Design*, 45:1469–1484, 2013. doi: 10.1016/j.cad.2013.06.010. 49

[100] T. R. Kannan and M. S. Shunmugam. Processing of 3D sheet metal components in STEP AP203 format. Part I: feature recognition system. *Intl Journal of Prod Research*, 47:941–964, 2009. doi: 10.1080/00207540701510055. 49

[101] Sean Tessier. Ontology based approach to enable feature interoperability between cad approach. Master's thesis, Georgia Institute of Technology, 2011. 49

[102] Christel Dartigues, Parisa Ghodous, Michael Gruninger, Denis Pallez, and Ram Sriram. Cad/capp integration using feature ontology. *Concurrent Engineering*, 15(2):237–249, 2007. 49

[103] Yogesh H. Kulkarni, Ravi K. Gupta, Anil Sahasrabudhe, Mukund Kale, and Alain Bernard. Leveraging feature information for defeaturing sheet metal feature-based cad part model. *Computer-Aided Design and Applications*, 0(0): 1–14, 0. doi: 10.1080/16864360.2016.1168238. 51

[104] Anton V. Mobley, Michael P. Carroll, and Scott A. Canann. An object oriented approach to geometry defeaturing for finite element meshing. In *7th International Meshing Roundtable*, pages 547–563, October 1998. URL `http://www.andrew.cmu.edu/user/sowen/abstracts/Mo512.html`. 61

[105] George Stiny and James Gips. Shape grammars and the generative specification of painting and sculpture. In *Segmentation of Buildings for 3DGeneralisation. In: Proceedings of the Workshop on generalisation and multiple representation , Leicester*, 1971. 69

[106] ALISON McKAY, Scott Chase, Kristina Shea, and Hau Hing Chau. Spatial grammar implementation: From theory to useable software. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 26(02):143–159, 2012. 69

[107] Peter Deak, Chris Reed, and Glenn Rowe. Cad grammars: Extending shape and graph grammars for spatial design modelling. *Design Computing and Cognition 06, Eindhoven, Netherlands*, 2006. 70

[108] Omer Akin and Hoda Moustapha. Formalizing generation and transformation in design. In JohnS. Gero, editor, *Design Computing and Cognition*,

pages 177–196. Springer Netherlands, 2004. ISBN 978-90-481-6650-3. doi: 10.1007/978-1-4020-2393-4_10. 70

[109] Hoda Moustapha. A formal representation for generation and transformation in design. Technical report, Carnegie Mellon University, 2004.

[110] Hoda Moustapha. *Architectural Explorations: A Formal Representation for the Generation and Transformation of Design Geometry*. PhD thesis, Carnegie Mellon University, 2005. 70, 72

[111] Alan Middleditch and Chris Reade. A kernel for geometric features. In *Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications*, SMA '97, pages 131–140, New York, NY, USA, 1997. ACM. ISBN 0-89791-946-7. doi: 10.1145/267734.267769. 71

[112] Gino Brunetti and Stephan Grimm. Feature ontologies for the explicit representation of shape semantics. *International Journal of Computer Applications in Technology*, 2003. 71

[113] Yoonhwan Woo and Sang-Hyun Kim. Protrusion recognition from solid model using orthogonal bounding factor. *Journal of Mechanical Science and Technology*, 28(5):1759–1764, 2014. doi: 10.1007/s12206-014-0322-0. 92

[114] Y-S Ma, G Chen, and G Thimm. Fine grain feature associations in collaborative design and manufacturing–a unified approach. In *Collaborative design and planning for digital manufacturing*, pages 71–97. Springer, 2009. 98

[115] Bih-Yaw Shih and Hiroshi Sakurai. Automated hexahedral mesh generation by swept volume decomposition and recomposition. In *Proc. 5th Int. Meshing Roundtable*, volume 96, pages 273–280. Citeseer, 1996. 98

[116] Y. Lu, R. Gadh, and T.J. Tautges. Feature based hex meshing methodology: feature recognition and volume decomposition. *Computer-Aided Design*, 33(3): 221–232, 2001. 99

[117] Yoonhwan Woo and Hiroshi Sakurai. Recognition of maximal features by volume decomposition. *Computer-aided Design*, 34:195–207, 2002. doi: 10.1016/S0010-4485(01)00080-X. 99, 101

[118] Yoonhwan Woo and Sang Hun Lee. Destructive modeling by volume decomposition and its applications. *ASME Design Engineering Technical conference*, 2003. 99, 101

[119] G Chen, Y. S. Ma, G Thimm, and S. H. Tang. Using cellular topology in a unified feature modeling scheme. *Computer-Aided Design and Applications*, 3:89–98, 2006. 99, 139

[120] Rafael Bidarra and Jos Carlos Teixeira. Intelligent form feature interaction management in a cellular modeling scheme. In *Symposium on solid modeling and applications*, pages 483–485, 1993. doi: 10.1145/164360.164543. 99, 101

[121] Rafael Bidarra, Maurice Dohmen, and Willem F Bronsvoort. Automatic detection of interactions in feature models. In *CD-ROM proceedings of the 1997 ASME design engineering technical conferences*, pages 14–17, 1997.

[122] Rafael Bidarra, Klaas Jan De Kraker, and Willem F. Bronsvoort. Representation and management of feature information in a cellular model. *Computer-aided Design*, 30:301–313, 1998. doi: 10.1016/S0010-4485(97)00070-5. 99, 101

[123] Jannis Klaas De Kraker. *Feature Conversion for Concurrent Engineering*. PhD thesis, Technische Universiteit Delft, 1998. 100

[124] Haiyan Wu and Gao Shuming. Automatic swept volume decomposition based on sweep directions extraction for hexahedral meshing. *23rd International Meshing Roundtable*, 82:136–148, 2014. doi: 10.1016/j.proeng.2014.10.379. 103

[125] Mark Keil and Jack Snoeyink. On the time bound for convex decomposition of simple polygons. *Proceedings of the 10th Canadian Conference on Computational Geometry, School of Computer Science, McGill University, Canada*, pages 54–55, 1998. 109, 110

[126] Mark Bayazit. Polygon decomposition, 2013. URL `http://mnbayazit.com/406/bayazit`. http://mnbayazit.com/406/bayazit (Last accessed Oct 2015). 110, 113, 116, 117

[127] Jairo Rocha and Rafael Bernardino. Singularities and regularities on line pictures via symmetrical trapezoids. In *Proceedings of the 4th International Conference on Document Analysis and Recognition*, ICDAR '97, pages 809–812, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7898-4. URL `http://dl.acm.org/citation.cfm?id=646270.684819`. 110

[128] Jairo Rocha. Polygon decomposition into singular and regular regions. Technical report, University of the Balearic Islands, 1999. 110

[129] Francisco de Moura Pinto and Carla Maria Dal Sasso Freitas. Fast medial axis transform for planar domains with general boundaries. pages 96–103, 2009. doi: 10.1109/SIBGRAPI.2009.21. URL `10.1109/SIBGRAPI.2009.21`. 121, 123

[130] H. Lipson and M. Shpitalni. On the Topology of Sheet Metal Parts. *Journal of Mechanical Design*, 120, 1998. doi: 10.1115/1.2826661. 132, 134

[131] S. H. Lee and H. S. Kim. Sheet modeling and thickening operations based on nonmanifold boundary representations. *Proceedings of DETC01, ASME2001 Design Engineering Technical Conferences*, 2001. 132

[132] Helen L. Lockett. A knowledge based manufacturing advisor for cad. 2005. URL `http://hdl.handle.net/1826/1116`. 134

[133] H. Lockett and M. Guenov. Graph based feature recognition for injection moulding based on a mid-surface approach. *Computer Aided Design*, 37:251–262, 2005. doi: 10.1016/j.cad.2004.06.010. 134

[134] Sang Hun Lee. Offsetting operations on non-manifold topological models. *Comput. Aided Des.*, 41(11):830–846, November 2009. ISSN 0010-4485. doi: 10.1016/j.cad.2009.05.001. 134

[135] Shriram Hegde. Aml710 computer aided design. URL `http://web.iitd.ac.in/~hegde/cad/lecture/L31_solidmodeling.pdf`. (Last accessed 20 July 2016). 134, 136, 144

[136] Ramesh Krishnamurti. *A Course on Geometric Modeling. Theory, Programming and Practice*. School of Architecture. Carnegie Mellon University, 2002. 136

[137] Kevin Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, 1986. 136

[138] Fujio Yamaguchi. *Computer-Aided Geometric Design: A Totally Four-Dimensional Approach*, chapter Fundamentals of Topology, page 358. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. ISBN 4431703403. 137

[139] Carlo H. Sequin. Generalized euler-poincare theorem. 2003. 137

[140] Grabcad. `https://grabcad.com/library/sheet-metal-parts-4`. Accessed: 2016-06-11. 169

[141] Unigraphics SDRC. Midsurface. `http://www.ugs.com/products/nx/ideas/`, Aug 2009. 201

[142] MSC. Dramatically accelerate mid-surface modeling to validation. Technical report, Smart Midsurface Software, 2015. 203

Publications

- **International Journals**:
  1. Jan 2017, *Dimension-Reduction Technique for Polygons*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Intl. Journal of Computer Aided Engineering and Technology, **Inderscience**.
  2. Jul 2016, *Computation of Midsurface by Feature-based Simplification-Abstraction-Decomposition*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Journal of Computing and Information Science in Engineering, **ASME**.
  3. Jul 2016, *Leveraging feature generalization and decomposition to compute a well-connected midsurface*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Engineering with Computers, **Springer**.
  4. Apr 2016, *Leveraging feature information for defeaturing sheet metal feature-based CAD part model*, Yogesh Kulkarni, Ravi Kumar Gupta, Anil Sahasrabudhe, Mukund Kale, Alain Bernard, CAD&A, **Taylor & Francis**.
  5. Apr 2015, *Topological Validation of Midsurface Computed from Sheet Metal Part*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Computer-Aided Design and Applications (CAD&A), **Taylor & Francis**.
- **International Conferences**:
  1. Jun 2015, *Defeaturing Sheet Metal Part Model based on Feature Information*, Yogesh Kulkarni, Ravi Kumar Gupta, Anil Sahasrabudhe, Mukund Kale, Alain Bernard Proceedings of CAD 15, **London, Taylor & Francis**.
  2. Dec 2014, *Formulating Midsurface using Shape Transformations of Form Features*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, All India Manuf. Tech. Design Research Conf. (AIMTDR), **IIT Guwahati**.
  3. Dec 2013, *Strategies for using feature information in model simplification*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Intl. Conf. on CAE, **IIT Madras**.
  4. May 2013, *Using Features for generation of Midsurface*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Intl. Conf. on Adv in Mech Eng, **COEP**.
- **National Conferences**:

1. Jan 2014, *Midcurves Generation Algorithm for Thin Polygons*, Yogesh Kulkarni, Anil Sahasrabudhe, Mukund Kale, Natl. Conf. on Emerging Trends in Eng and Science , **Asansol, India**.

Appendices

Chapter A

Survey Responses

- **Why do we need to perform model simplification in the era of (almost) infinite computation power?**:

  Model Simplification is essential even when computation power has increased dramatically ((87)), because:

  – more complex problems can be solved

  – more design iterations can be performed

- **Which Domain needs Midsurface the most?**:

  Thin-walled components are widely used in aerospace, automotive and many other industries (Figure A.1) due to their advantages of material/cost/energy saving and easy manufacturability. Injection Moulding parts are most complex due to draft and have most demand for better Midsurface. Here even surface does not get generated, then connections. Sheet Metal parts are relatively easy, create surfaces but fail in connections. So it is better to start with Sheet Metal parts and then attempt generic or injection molding parts

Figure A.1: Thin Wall Applications (Source: Autodesk ((28)))

- **Which input format is most suitable?**:

  Feature based CAD (Brep) models are far richer has they application specific sub-

volumes apart from geometry and topology. This research does not deal with Mesh, Voxel, plain Brep, CSG model. Features can be live via API or through some neutral export format ((49)).

- **What are advantages of Midsurface wrt Shell Elements?**:
  - Shell Elements are more efficient than 3D elements for Thin Wall (Figure A.2)
  - Experimentation with just thickness can be done, for 3D elements remeshing would be needed
  - For 3D elements minimum 3 layers would be needed which is not feasible for really thin portions
  - Solid elements may show locking under bending dominated loading

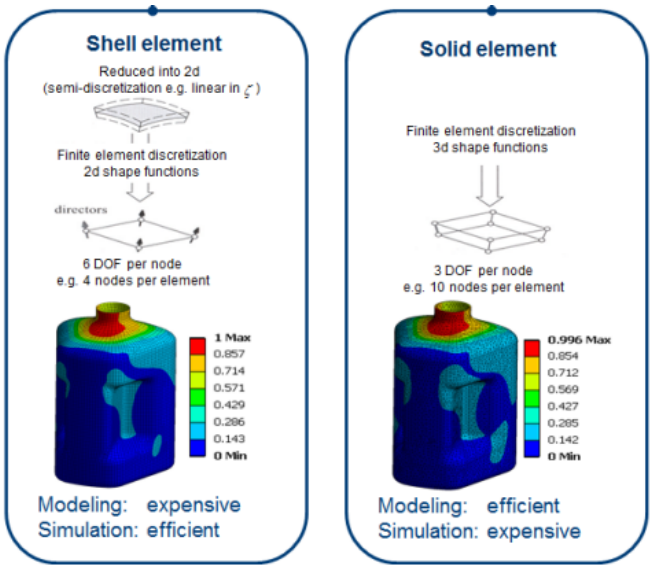


Figure A.2: Comparative Analysis of Shell Solid Elements (Source: Digital Eng ((4)))

- **Why compute surface at the Mid and not just use one side?**:

  For thin-walled parts, instead of computing Midsurface, the obvious choice is to use either of the top/bottom surfaces to place the shell elements. But these surfaces are not good choices and we have to go for the Midsurface as it truly represents the part. In case of assemblies or connections midsurfaces connect better across the parts. Also thickness-data/material is represented well as it distributed on both sides in case of Midsurface. For thin-walled parts, the choice of surface can be either outer, inner or somewhere in the middle. In the first two cases, surfaces are ready and no new computation is needed. But still midsurface seems to be a preferred option.

  - **Geometric Consideration**:

    Defining a mesh of thin shell elements on a part's Midsurface may yield better results than defining a mesh on either the part's outer or inner surfaces. This is because the mesh on the Midsurface gives a closer approximation of the volume of the part ((141)).

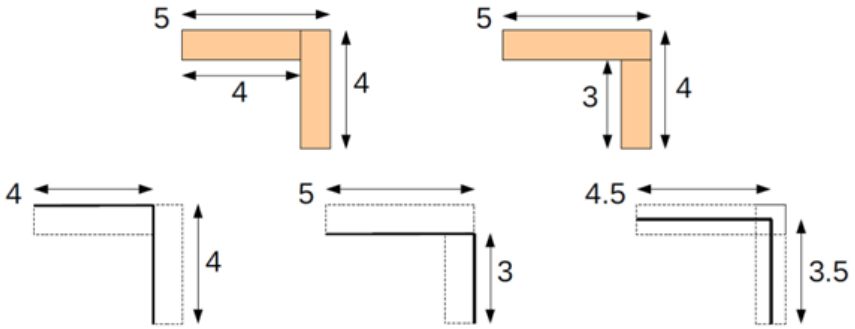


Figure A.3: Need for Midsurface Compared to One Side (Source: Smit ((29)))

- **Analysis Consideration**: If you choose a surface for the shell mesh placement that isn't at the Midsurface of the part, the analytical part will differ from the CAD part by $1/2$ a wall thickness everywhere. In some areas, your part may analyze as larger than you expected, in others, smaller. It is comforting to note that the higher the part's aspect ratio or the better candidate a part is for a shell mesh, the less shell surface placement matters. In the grand scheme of things, on a system as large as that garbage truck body shown earlier, $1/2$ wall thicknesses either way will not impact the results in a measurable way. However, as a part approaches the gray zone where it could be modeled as shells or solids, Midsurface placement of the shell becomes important ((37)).

- **What are the issues with prominent medial generation methods?**:
  - MAT: unsuitable as it creates branches, needs pruning. This can be used in 2D profile case along with Parametric and/or tringularization (chordal method) for connections.
  - MA: Face pairing is complex
  - Feature based: So far Sweep based (Extrude, Revolve, Sweep) and primitives like parallelepiped, cylinder, wedge, Pad, Pocket as individual features have been tried. But a generic method, taking care of interactions is missing.
  - Errors: Gaps, missing surfaces, overlapping surfaces etc. (Fig. A.4)

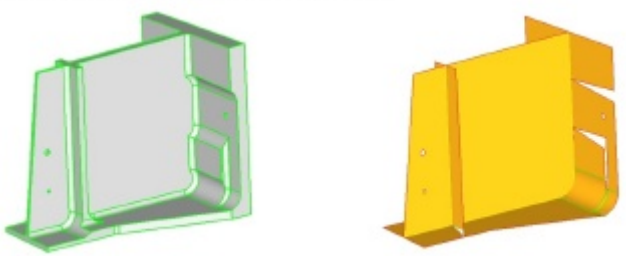


Figure A.4: Midsurface Gaps (Source: Austreng ((30)))

  - Commercial:

- For Sheet Metal, majority of surfaces come out fine. Only problems are at the connections, which we do on Mesh. Tubular structures (such as Motorcycle chassis) have quite a few connection problems.
- For plastic parts, problems are many. Variable thickness, steps, small but elongated features, are main causes of not getting good Midsurface. Automatic Midsurface is almost not useful.

Relevant findings/quotes by various researchers are:

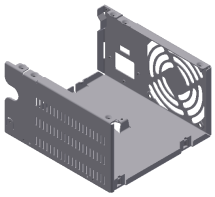
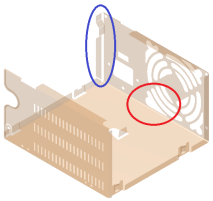
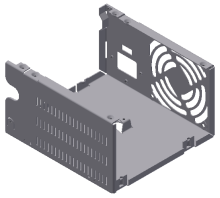
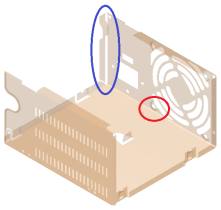
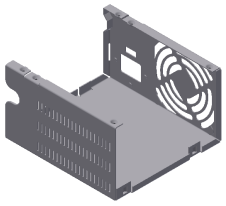
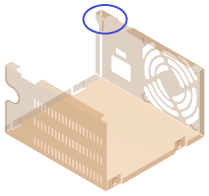
- *"There is a definite need for a dimensional reduction capability that is more powerful and easier to use than those currently available in the market. Such a capability should deliver an automated scheme for handling cases that have traditionally caused problems for algorithms in this field." - Stanley Felix, 2010* ((69))
- *"Much of research is yet to be done, use of symmetry, various features, various abstractions are not yet handled." - Matt Smit, 2011* ((29))
- *"Valid mid-surfaces may not be abstracted, and mid-surfaces which are abstracted may not be useful in practice as solid models become increasingly complex."- Yoonhwan Woo, 2013 ((3))*
- *"Mid-surface repair is tedious and manual" - MSC at the launch of semi-automatic 'Smart Midsurface', 2015 ((142))*

Chapter B

Defeaturing Threshold

Following test cases show the effect of defeaturing on the quality of the midsurface. Size threshold used here is certain percentage of the summation of face-areas of all the faces in the original CAD model.

1. Threshold (D) 3% of the total part size

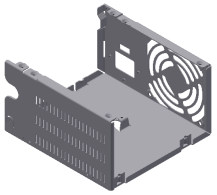
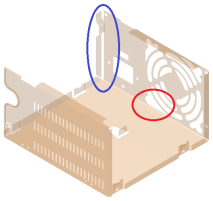
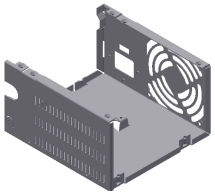
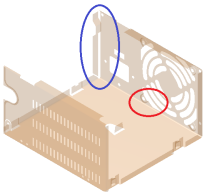
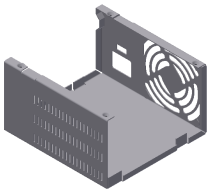
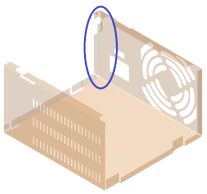
	Model	Midsurface	Explanation
Original input			Gaps in the midsurface. Two of the gaps are marked.
Removing sheet metal features less than threshold			Although the number of gaps have reduced, but the gaps between the surface patches is still seen.
Removing generic CAD features less than threshold			Some of the gaps at the top portion are filled and the output is a bit better-connected midsurface.

Effectiveness of Smarf with 3% threshold by Eqn. 4.1 is:

Qty	**Input**	**Phase I**	**Output**
Faces	1434	610	327
Suppressed		60	30

$$pR = (1 - \frac{697}{833}) \times 100 = 16\%$$

2. Threshold (D) 5% of the total part size

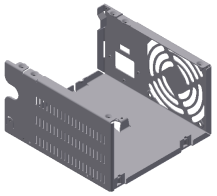
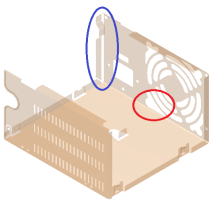
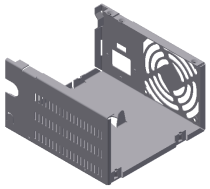
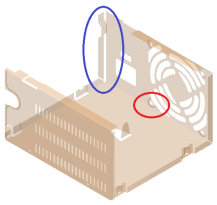
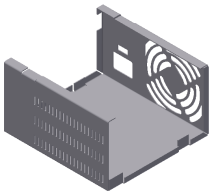
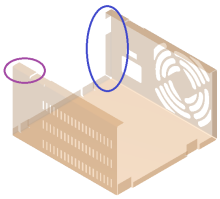
	Model	Midsurface	Explanation
Original input			Gaps in the midsurface. Two of the gaps are marked.
Removing sheet metal features less than threshold			Although the number of missing gaps in the midsurface has reduced, but the gaps between the surface patches is still seen.
Removing generic CAD features less than threshold			Most of the gaps are filled and the output is a better-connected midsurface. It retains all the necessary features adequately "representing" the gross shape.

Effectiveness of Smarf with 5% threshold by Eqn. 4.1 is:

Qty	**Input**	**Phase I**	**Output**
Faces	833	772	617
Suppressed		7	40

$$pR = (1 - \frac{697}{833}) \times 100 = 26\%$$

3. Threshold (D) 10% of the total part size

Model	Midsurface	Explanation
Original input		 Gaps in the midsurface. Two of the gaps are marked.
Removing sheet metal features less than threshold		 Although the number of missing gaps in the midsurface has reduced, but the gaps between the surface patches is still seen.
Removing generic CAD features less than threshold		 Most of the gaps are filled. Removal of purple feature could be the domain decision. It retains all the necessary features adequately "representing" the gross shape.

Effectiveness of Smarf with 10% threshold by Eqn. 4.1 is:

Qty	**Input**	**Phase I**	**Output**
Faces	833	715	522
Suppressed		17	48

$$pR = (1 - \frac{522}{833}) \times 100 = 37\%$$

The choice of threshold can be set to such a value where midsurface output is well-connected. In the test-cases shown above 10% appears to be the appropriate value. Increasing the threshold to still higher value results in a well-connected midsurface but it loses the shape characteristics which need to be maintained.

Chapter C

ICE Schema

- *Regulators* are in **bold**, $\mathcal{R}$, represents a unit of action
- *shapes* are in *lowercase*, say, *circle, profile*
- Prefix to a *Regulator* is its *category*, like,
  Δ : transformations
  Φ : constraints
  Ψ : hierarchies
  Π : topologies
  Ξ : variations
  Ω : operations
- Superscripted^{suffixes} indicate regulator *subtype*, for instance, ΔC^p and ΔC^e respectively specify parabolic and elliptical curve regulators
- Numerical^{suffixes} denote the dimension of the *Regulator*, for instance, ΔM^0 , ΔM^1 , and ΔM^2, respectively represent a mirror point (0-dimensions), a mirror line (1-dimension) and a mirror plane (2-dimensions)
- Subscripted$_{suffixes}$ for *Regulators, shapes* represent as indices; for example ΔT_1, and ΔT_2 are two different instances of *Translation Regulators* used in the same configuration.
- *Regulators* can be generative or non-generative. Generative regulators (depicted by the presence of the "n" parameter) take an input shape and create output shapes, while non-generative regulators act on the input shape.
- $\Delta\mathbf{T}(\bar{s})^{<0><1><2>}$ is a discrete application generating disjoint points, while $\Delta\mathbf{T}(\bar{s})^{<0,1,2>}$ is a continuous application generating a line.
- Key-points (e : endpoint, m : midpoint, s : start-point) To access key-points (such as the midpoint): $shape_A\langle m_1\rangle$, length by $shape_A\langle l\rangle$

Although entities like *point*, *line* are present in ICE but entities and features pertinent to Mechanical CAD are not present. Also treatment given to primitives is different in the proposed approach. The additions to ICE are:

- **Entities** : CAD objects like *profile*, *sketch* etc.
- **Regulator**: Changed the definition to include *guide* (directrix) and removed hard coded $\bar{t}, d$
- **Class Hierarchy** : Inheritance *derived::parent* relationships between entities. Operations can be defined in terms of the *parent* entities so that they are applicable to *Derived* classes as well.
  $derived$ :: $base$: subclass relationship
  $|$: logical OR
  & : logical AND
- **Form Features** : Definitions of variety of form feature and operations like patterning etc.