Formulating Midsurface using Shape Transformations of Form Features

Yogesh H Kulkarni[1], Anil Sahasrabudhe[2], Mukund Kale[3]

[1]* College of Engineering Pune, kulkarniyh12.mech@coep.ac.in

[2]College of Engineering Pune

[3]Siemens PLM Pune

Abstract

Shapes modelled using Computer Aided Design (CAD) applications are used in downstream applications like, manufacturing (Computer Aided Manufacturing, CAM), Analysis (Computer AidedEngineering, CAE) etc. Use of form features is prevalent in the CAD applications, but their leveraging in the downstream applications is not very common, especially in the CAE applications.

The initial phase of design demands for quick analysis of the model. Here, CAD models are oftensimplified by removing the irrelevant features (de-featuring) and by idealizing solids to surfaces or curves(dimension reduction), so that the CAE analysis gets performed with lesser resources and time. MidsurfaceExtraction is one of the ways of dimension reduction where thin-walled portions of a solid areidealized to surfaces lying midway.

This paper presents a novel representation scheme (called ABLE) for CAD entities and operators including formfeatures which is then leveraged to define the algorithm for extracting Midsurface.

Keywords: CAD, Spatial Grammar, Form Features, Sheet Metal Features, CAE, Midsurface

1 Introduction

CAD applications are widely popular in product design. They aid the design process by presenting easy-to-use ways of defining shapes, attributes, relationships etc. Shapes modelled in CAD can then be used in various downstream applications like drafting, analysis, manufacturing etc. Many commercial CAD applications provide *Design-by-Features* approach. Features not only carry shape information (geometry, topology) but also embed meta-information based on the application's need. Features also reflect terminologies used in the application, thus making them intuitive to use. But this has given rise to various feature-schemes not only in different CAD applications but also in various environments present within the same CAD application. Shape that a feature represents could be similar but its featurenomenclature, usage, could be different in different environments-applications. Features like *Box*, *Pad*, *Protrusion*, *Extrude*, appear different in nomenclature, but they all could be representing the same shape-operation. This diversity of vocabulary creates problems in learning new CAD systems, interoperability between different CAD applications, and also in development of functionality for downstream applications. This is evident in relatively lesser usage of features in the downstream applications, especially, CAE. Instead of plethora of feature nomenclatures, a neutral-standardized internal representation would be very useful. Functionality developed on top of such representation will have advantage of applicability over variety of feature representations present in different CAD applications.

This paper presents an idea of formalizing form features in terms of Spatial Grammar notations and demonstrates its use in developing algorithms on top of it, for downstream applications, like Midsurface for CAE.

2 Related Work

Following subsections explore salient work done so far in the areas relevant to the topic, like, Spatial Grammars, Form Feature Representation and Midsurface Representation.

2.1 Spatial Grammars

Spatial Grammars is a general term which encompasses shape definition Grammars, like Graph Grammars, Shape Grammars, Set Grammars, etc. Aim of Spatial Grammars is to bring formalism through terse but expressive definitions, validations and generation of new evolutionary shapes. Although primarily used for generative designs in architecture, Hoisl et al.[3] proposed a Spatial Grammar based system for implementation in CAD, for generative solutions. CAD Grammars proposed by Deak et al. [2] combined Shape and Graph Grammars to be more useful in Design, Modelling and Manufacturing.In general there has been limited success to the usage of Spatial Grammars in CAD-CAE due to its complexity of formulation and non-intuitive user interfaces..

2.2 Form Feature Representation

A generally accepted definition of feature is that it represents shape as well as functionality significant to particular product life-cycle phase. Features through use of taxonomy, semantics and ontology bring formalized knowledge representation which can be leveraged to address problems such as interoperability between CAD system and developing downstream applications such as CAE, CAM etc. [1].

Use of Spatial Grammar notations to represent scheme of generalized form features is relatively new and is not widely utilized in the commercial CAD applications. This paper formalizes an approach to such representation and later uses it to develop Midsurface which is an important idealization methodin CAE.

2.3 Midsurface Representation

Midsurface is an abstraction of thin-walled portions of solid intoa set of connected surfaces that lie midway and carry thickness information needed to define 2D shell elements (Figure 1). There are various techniques to extract Midsurface mentioned in academic world and available in commercial CAD-CAE applications. Predominant among those techniques are, Medial Axis Transform (MAT) and Midsurface Abstraction (MA) method.

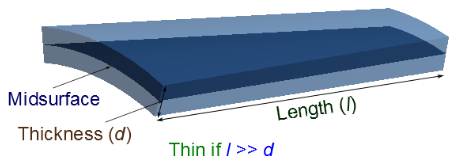


Figure 1: Midsurface

Most of the Midsurface extraction approaches are based on the final shape, represented typically by Boundary Representation (Brep) solid and does not use feature information.

This paper demonstrates use of form features defined in terms of Spatial Grammars notations in formulating Midsurface.

3 New Form Features Representation

Form Features come in different flavors in different CAD applications. Even within a same application, similar looking features are presented in different ways.

Hoisl [3] mentioned that a limited set of shapes can be generated using *Sweep* in a generic manner. A *Sweep* is an operation where moving a shape (called *generator*) along a trajectory (called *guide curve*) creates variety of shapes. *Sweep*, by its strict meaning, is limited by the use of a single *profile* only. A more generic operation is*Loft*, where multiple *profiles* are joined together, along a *guide curve*.

Representation scheme used in this paper is loosely based upon Interactive Configuration Exploration (ICE) scheme developed by Moustapha [5]. Although some of the fundamental entities and syntaxes are borrowed from ICE, we have enhanced it to suit Mechanical CAD application.

3.1 ICE scheme

"*The ICE notation is formalism for describing shapes and configurations, by means of their generative and relational structures*" [5]. 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 generic *Regulator* is represented as

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

For example,

$$\Delta\,\boldsymbol{A}_2^{T,1}[\{\bar{p}, line, n\}(shape)]$$

Where,

$\boldsymbol{\Delta}$: Transformation category (type)
$\boldsymbol{A}$: Affine Transformation Regulator (type)
^{T}: Subtype:Translation (type)
^{1}: dimensionality of output (integer)
$_{2}$: 2nd instance
$\bar{p}$: Position (point)
$line$: Linear guide (curve)
n: number, copies, scaling factor etc. (float)
$shape$:target (shape)

The ICE notationcan be mapped to grammar-formalism presented in A → B manner. Shape in the (…) bracket is the input to the rule and can be mapped to LHS (A) for the matching condition. On this LHS, transformation rule (A → B) is applied, using specified-arguments in the {…}, to generate RHS (B).

3.2 New enhancements to ICE

Although entities like *point*, *line* are present in ICE entities and features pertinent to Mechanical CAD are not present. Our new additions to ICE are:

- **Entities**: CAD objects like *profile*, *sketch* etc.
- **Class Hierarchy**: Inheritance *child::parent* relationships between entities. Operations can be defined in terms of the parent entities so that they are applicable to child classes as well.
- **Form Features**: Definitions of variety of form features and operations like patterning etc.

3.2.1. Entities (*E*):

The fundamental primitive is the*point*. All other geometric entities are directly or indirectly defined in terms of points.

Shape (*shape*): The base class. All other entitiesare directly or indirectly derived from it.

Point (*point::shape*): The fundamental primitive expressed as$\bar{p}$.

Line (*line::curve*): Defined by two points ($\bar{s_1}$and$\bar{s_2}$), and is expressed as Loft of $\bar{s_1}$ along $line$ up to$\bar{s_2}$

$$\Omega\, \boldsymbol{L}^{T,1}[\{0, line, 0\}(\bar{s_1})^{<1>}]$$

Curve (*curve::shape*): A generic entity modelled in terms of *n* points, and expressed as

$$\Omega\, \boldsymbol{L}^{C,1}\left[\{0,0, C_{0|1|2}\}(\bar{s})^{<1-n>}\right]$$

Profile (*profile::shape*): Collection of *n* connected curve segments and is expressed as

$$\Pi\, \boldsymbol{C}^{P,1}\left[\{0,0, C_{0|1|2}\}(curve)^{<1-n>}\right]$$

Sketch (*sketch::shape*): Collection of *profiles*, first outer and rest inner and is expressed as

$$\Pi\, \boldsymbol{C}^{S,1}[\{\}(profile)^{<1><2-n>}]$$

Surface (*surface::shape*): Modelled using collection of *U,Vcurves* and is expressed as

$$\Omega\, \boldsymbol{L}^{F,2}\left[\{0, (curve)^{<1-n>}, C_{0|1|2}\}(curve)^{<1-m>}\right]$$

Solid (*solid::shape*): Modelled using generic *surfaces* and expressed as

$$\Pi\, \boldsymbol{C}^{R,3}\left[\{0,0, C_{0|1|2}\}(surface)^{<1-n>}\right]$$

3.2.2 Affine Transformation Operators (*A*):

Affine Transformations use matrix multiplications and can be performed on entities of various dimensions, like *point* (0), *curves* (1), *surface* (2) and *solid* (3). All these are generically clubbed together under ***A*** with *subtypes* as *T,R,*and *S* for Translation, Rotation and Scaling respectively.

Translation moves *shape*, along *line* and is expressed as

$$\Delta\, \boldsymbol{A}^{T,0|1|2|3}[\{0, line, 0\}(shape)]$$

Rotation rotates *shape*, by along *arc*, about point $\bar{p}$and is expressed as

$$\Delta\, \boldsymbol{A}^{R,0|1|2|3}[\{0, arc, 0\}(shape)]$$

Scaling scales *shape*, by factor *f*, about an $axis$ and is expressed as

$$\Delta\, \boldsymbol{A}^{S,0|1|2|3}[\{axis, f, 0\}(shape)]$$

3.2.3. Copy commands like *Pattern* can be modelled as Affine Transformations with n copies.

Linear Pattern copies *shape* linearly, *n-1* times and is expressed as

$$\Delta\, \boldsymbol{A}^{T,0|1|2|3}[\{0, line, n\}(shape)]$$

Circular Pattern copies *shape* circularly, *n-1* times and is expressed as

$$\Delta\, \boldsymbol{A}^{R,0|1|2|3}[\{0, arc, n\}(shape)]$$

3.2.4. Boolean Operators (*B*):

Boolean operations are also generic in the sense that they can apply 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 $shapes(shapes_{1-k})$ into master $shape(shape_0)$and is expressed as

$$\Omega\, \boldsymbol{B}^{U,1|2|3}[\{\}(shapes_{0-k})]$$

Difference removes combination of all the tool $shapes(shapes_{1-k})$ from master $shape(shape_0)$and is expressed as

$$\Omega\, \boldsymbol{B}^{D,1|2|3}[\{\}(shapes_{0-k})]$$

Intersection keeps only common portion of all the $shapes(shapes_{0-k})$into master $shape(shape_0)$ and is expressed as

$$\Omega\, \boldsymbol{B}^{I,1|2|3}[\{\}(shapes_{0-k})]$$

3.2.5. Loft Operators (*L*):

Loft is a generic operator (Figure 2) capable of generating most of the basic shapes. It joins *profiles* along a *guide curve*.

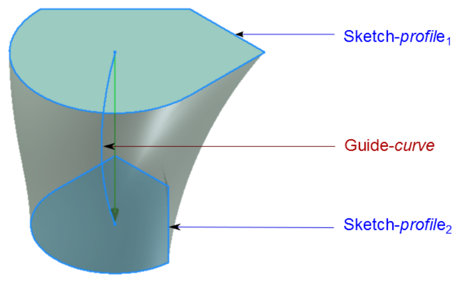


Figure 2: Generic Loft feature

Represented as:

Continuity options like for connectedness, for tangency and 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. Some form features also specify a draftangle for tapering sides. This can be modelled as Loft between two *profiles*, where the second *profile* is offset-ed inside-or-outside. 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.

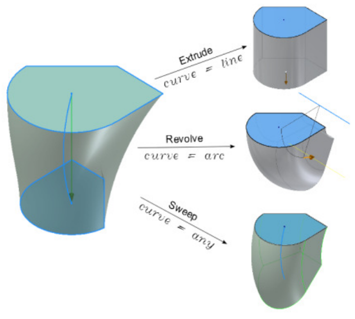


Figure 3: Extrude, Revolve and Sweep in terms of Loft

Loft can manifest itself in different forms (Figure 3) as elaborated below:

Extrude without draft is denoted by *EnDsubtype*, has single *sketch*, swept along a *line* and is expressed as

Extrude with draft is denoted by *EwDsubtype*, has two *sketches* between which loft is made along a *line* and is expressed as

Similarly,*Revolve*, *Sweep*, *Loft*, with or without draft, can be expressed. Primitive shapes are further specializations of ***L*** operators like *Extrude* and *Revolve*. By changing shape of the *profile* and *curve* one can get different standard primitives (Figure 4)

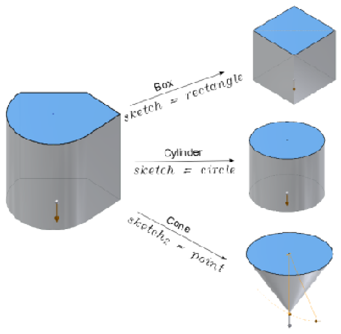


Figure 4: Box, Cylinder, and Cone in terms of Extrude-Loft

As demonstrated above, most of the form features used in CAD application can be modelled using three operators, viz. Affine Transformation (***A***), Boolean (***B***) and Loft(***L***) and set of Entities(***E***).

This representation is called as ***ABLE***. Other features like *Shell*, *Fillet*, and *Chamfer* can also be formulated using ***ABLE***.

4 Representing Sheet Metal features using *ABLE*

Sheet Metal parts aremodelled either by using *sketch* based features or by preparing a solid model first and then making it hollow using *Shell* operation. The first approach is elaborated below whereas the latter can be treated as subtraction of an offset-ed tool body from the target body.

Face-Wall-Base Flange can be represented by *Extrude* of a given *profile*.

Rib is a triangular *profile* extruded.

Hole-Cutout-Slot is a negative *boolean* of a tool body which can be represented by *Extrude* of a respective *profile*.

Lance-Louver is a combination of cut first, then addition of a *Sweep* with respective *profiles*.

Bend is a *Sweep* of a Rectangular *profile* of edge length. One can add subtype based on the type of relief provided.

Draw-Coin is a specialized *Loft*, with draft, with *guide curve* having *fillets* at the start and end.

Following section demonstrates how ***ABLE*** can be used to express the Midsurface operator.

5 Midsurface using *ABLE*

In a feature tree, at each feature level, the operands (especially the tool bodies) are relatively simpler, thus making Midsurface Extraction more deterministic [4].

Use of Spatial Grammar has made it possible to come up with a terse generic representation (as demonstrated by ***ABLE***) so that algorithms need not have to work on all but just few generic operators.

For features based on ***L*** operator, feature parameters such as *sketches-profiles*, *curves* etc. are extracted first and then, based on the rules specified below, Midsurface is generated.

Midcurve is a set of *curves*lying midway of 2D *profiles*and is expressed as

Midsurface rules differ based on the relative sizes of *profiles* and *curve*.

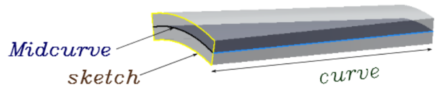


Figure 5: Sketch is smaller than Curve

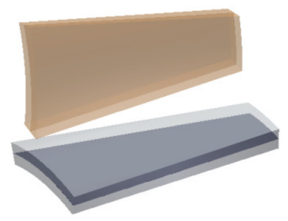
Thin Sketch: If *sketch* is very small compared to *curve* (Figure 5), then midcurve is extracted from the sketch (as mentioned above) and Lofted with same feature parameters as that of ***L***. This rule is expressed as

Thin Loft: If *sketch*is very big compared to *curve* , then *midcurve* is not extracted but the *sketch* itself is lofted along half of the *curve*. This rule is expressed as

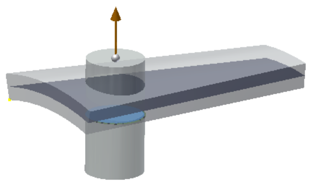
Thick Sketch: If *sketch*is comparable in size to *curve* , then it is a thick shape and Midsurface is not generated.

For ***B*** operators rules are devised depending on where the target and tools have Midsurface extracted already.

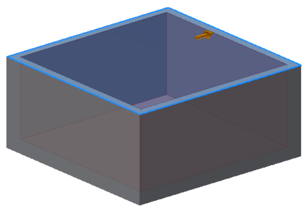
Union: If the target and the tool bodies have Midsurfaces, extend-trim them to join at the common intersections.



Difference-Thin: If the target has Midsurface then irrespective of tool bodies being thick or thin, they are subtracted.



Difference-Thick: If target and tool bodies are thick and are in *Shell* like situation, midcurves of combined profiles are calculated and*Lofted.*



6 Conclusion

This paper proposes a new representation, using Spatial Grammars notations, called ***ABLE***,to provide concise definitions of form features, and application specific features like Sheet Metal features. It shows how a generic feature, like *Loft*, can be manifested into variety of form features.Using just a few operations, like *Loft* (***L***), *Booleans* (***B***) and *AffineTransformations* (***A***) along with entities, a CAD model can be built.It also presents how a feature based algorithm, e.g. the Midsurface Extraction algorithm, can be effectively devised using the neutral definitions provided by ***ABLE***. This algorithm thus becomes portable to any CAD application which can provide feature-entities representations in ***ABLE*** format.

References

[1] R. Bidarra and W.F. Bronsvoort,*Semantic feature modelling*, Computer-Aided Design, 32(3):201–225, March 2000.

[2] 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.

[3] Frank Rainer Hoisl,*Visual, Interactive 3D Spatial Grammars in CAD for Computational Design Synthesis*, PhD thesis, TU Munchen, 2012.

[4] Yogesh Kulkarni, Anil Sahasrabudhe, and Mukund Kale, *Strategies for using feature information in model simplification*, In Proceedings of International conference on Computer Aided Engineering, IIT Madras, 2013.

[5] Hoda Moustapha,*Architectural Explorations: A Formal Representation for the Generation and Transformation of Design Geometry*, PhD thesis, 2005.