Generating CAD Parametric Features Based on Topology Optimization Results

William R. Blattman
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation
Blattman, William R., "Generating CAD Parametric Features Based on Topology Optimization Results" (2008). All Theses and Dissertations. 1371.
https://scholarsarchive.byu.edu/etd/1371

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
GENERATING CAD PARAMETRIC FEATURES BASED ON TOPOLOGY OPTIMIZATION RESULTS

by

William Robert Blattman

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

April 2008
This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date

C. Greg Jensen, Chair

Date

Christopher A. Mattson

Date

Gary R. Lawlor
BRIGHAM YOUNG UNIVERSITY

As chair of the candidate’s graduate committee, I have read the thesis of William R. Blattman in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

C. Greg Jensen
Chair, Graduate Committee

Accepted for the Department

Matthew R. Jones
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology
ABSTRACT

GENERATING CAD PARAMETRIC FEATURES BASED ON TOPOLOGY OPTIMIZATION RESULTS

William R. Blattman
Department of Mechanical Engineering
Master of Science

Shape optimization has become an important tool in industry to minimize weight and generate new designs. At the same time, companies are turning to CAD-centric design strategies where robust parametric CAD models are used to generate new designs and part-families of current designs, as well as the tooling and manufacturing procedures. However, due to its complexity, the optimal topology results are often discarded or recreated by hand into a CAD model. From a design standpoint, the results can be improved with the use of manufacturing constraints on the shape optimization process. These constraints improve the manufacturability based on common manufacturing practices. Even with these improvements, the process of converting topology results to CAD can cost substantial amounts of time and money.
This thesis proposes a method of semi-automatically recognizing the voids, created during the shape optimization process, with parametric features based on CAD geometry construction. These parametric features are based on sets of cross-sectional shapes and spine rules to create solid objects. These features are then sent to the CAD part file via programming APIs that exist in the software packages.

By recognizing features usable to the CAD systems, the voids can be characterized in the CAD model using robust dimensional constraints. This allows for the CAD model approximation to represent the topology optimization results with dimensional values from simpler shapes. Size optimization can then be applied to optimize the approximating model and regain any fidelity loss in the analytic model.

Test cases created with and without manufacturing constraints show considerable promise in a proof-of-concept scenario. These tests utilize the topology optimization software HyperMesh from Altair and the CAD package NX 4.0 from Siemens (formerly UGS). The voids from shape optimization in these tests are recognized inside of HyperMesh, fit with a simple parametric feature, and created in the part model using the Open C API in NX.
ACKNOWLEDGMENTS

My advisor Dr. Jensen, my graduate committee, the ParaCAD lab, and most of all my loving wife Jessica.
TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................................................... xi

LIST OF FIGURES ........................................................................................................................................ xii

1 Introduction ............................................................................................................................................... 1
  1.1 Problem Statement ........................................................................................................................... 2
  1.2 Thesis Objectives ............................................................................................................................ 3
  1.3 Delimitations of the Problem ............................................................................................................ 3
  1.4 Anticipated Benefits ......................................................................................................................... 4
  1.5 Conclusions/Recommendations ......................................................................................................... 4

2 Literature Review / Background ....................................................................................................... 5
  2.1 Topology Optimization ....................................................................................................................... 5
    2.1.1 Homogenization ......................................................................................................................... 6
    2.1.2 Hard/Soft Kill Method ............................................................................................................... 7
    2.1.3 Manufacturing Constraints ....................................................................................................... 7
    2.1.4 Commercial Options ............................................................................................................... 8
    2.1.5 Existence of Voids .................................................................................................................... 9
  2.2 Shape Fitting/Recognition ............................................................................................................... 10
  2.3 Feature Mathematics ....................................................................................................................... 12
  2.4 Advanced Parametrics ...................................................................................................................... 15
  2.5 Size Optimization ........................................................................................................................... 18
  2.6 API Programming ........................................................................................................................... 20
LIST OF TABLES

Table 5-1: Volumes for torsion bar topology results and approximations .................... 58
Table 5-2: Volumes for cube topology results and approximations................................ 64
LIST OF FIGURES

Figure 2-1 A non-regular set that fails to match the closure of the interior .......... 9
Figure 2-2 Inserting void B into a set A and showing their regular set ............. 10
Figure 2-3 Cross-sectional curves, spine curve and associated feature .......... 14
Figure 2-4 Complex features created using sets of cross-sections and spine rules. .... 15
Figure 2-5 Two different parameterizations for a free-standing vertical column .... 17
Figure 2-6 Two fidelity levels of a part file controlled by a Boolean parameter ....... 18
Figure 3-1 Two interpretations of a complex void based on designer intent .......... 26
Figure 3-2 Manual void selection using a ‘by window’ selection tool .............. 28
Figure 3-3 Void selection and segregation of a complex void ................. 29
Figure 3-4 Searchable area highlighted based on void segregation ................. 29
Figure 3-5 Pareto front of optimal solutions for minimizing error and complexity .... 31
Figure 3-6 Different shape complexity levels based on different user inputs .......... 32
Figure 3-7 Potential problems with minor changes in topology ................. 33
Figure 3-8 Changes in geometry due to size optimization ......... 34
Figure 3-9 Void interpreted with a single cross-section and a linear spine rule .... 35
Figure 3-10 Void with a single cross-section and multiple spine rules .......... 35
Figure 4-1 By window selection method for selecting nodes in HyperMesh ......... 39
Figure 4-2 By window selection results .......... 39
Figure 4-3 Example of nodal selection inside of HyperMesh 8.0 .......... 40
Figure 4-4 Least squares algorithm for fitting a circle to the cloud of points ......... 42
Figure 5-1 Simple torsion load on a cantilevered beam ................. 48
1 Introduction

Shape optimization is becoming more important as industry attempts to minimize weight while still maintaining a sufficiently strong and safe design. Shape or topology optimization utilizes new methodology developed in the last 20 years to discover the optimal shape within a given design envelope. Commercially, there exists software to perform this optimization problem on finite element models (FEM). Their resultant optimal topology, however; only exists as discrete FEM nodes and elements or as smoothed generic geometry, such as IGES or STL. Utilizing this newly discovered topology is difficult and often needs to be interpreted by a designer and recreated in the CAD model.

Robust parametric CAD models are utilized in every phase of the design process. Beginning in the conceptual design, parametric CAD models are used to quickly create and evaluate new designs and ideas. Once the designs are formalized, entire part-families can be developed though the use of robust parametric strategies. A part-family, in this circumstance, is an entire library of related designs represented by a single parametric model. These same CAD models are used throughout the life of the part. They are even used in tool designs and manufacturing of the part. Thus, the CAD model serves as a starting point for concept evaluation, various types of analyses, manufacturing changes and optimization procedures (such as shape and size optimization).
1.1 Problem Statement

The use of topology optimization in industry is increasing, but the usability of optimized shape models is not. Some have noted that one of the major inadequacies of topology optimization in design is, “the interpretation of topology optimization results and the lack of automated information extraction from design database for subsequent applications.” [28]

 Efforts have been made to improve the smoothness of the optimal shape and, to an extent, export the shape as generic geometry. But, the usability of the optimal shape models has not increased substantially. Currently, no commercial CAD system incorporates a strong link between topology optimization results and their robust parametric models. The only tangible link uses exported topology results and imports them into a CAD model as relatively unusable geometric features. They have little or no direct (automated or semi-automated) capacity for converting tessellated or smoothed optimized topology models into manufacturable, parametric models.

 The current sequence employed by many companies to implement information from a topology optimization model into the CAD parametric design involves a designer manually interpreting features from the optimized topology model and creating them by hand into an existing CAD part file. The engineer, in this scenario, is responsible for both the choice of the CAD feature, the manufacturability of the feature, and the fit/location. This process can cost substantial amounts of money and numerous work hours.
1.2 Thesis Objectives

The purpose of this thesis is to create an architecture that eliminates much of the work of new feature selection and fit/location from the topology results while allowing the designer the flexibility to take into consideration engineering criteria. This thesis also demonstrates one example of implementing the architecture of passing information from newly discovered features to the CAD model. This is intended to be a CAD-centric approach. The topology optimization results are used to update the CAD model. The model is the starting point for all future operations, and can be done in a generic manner that is applied in various environments and systems.

1.3 Delimitations of the Problem

The scope of this thesis is narrowed to the creation of a single architecture that demonstrates the transfer of information from the optimal topology model to the CAD model. A demonstration of this architecture is shown with a simple implementation.

The scope of defining parametric CAD geometry with topology results is restricted to the recognition of three-dimensional shapes constructed from simple two-dimensional, prismatic cross-sections consisting of straight lines and circles (no arcs). These shapes are limited to extruded features (which rely on two-dimensional cross-section) and will only be shown for those features that go completely through the model (only thru-cuts). To validate the method, three case studies are employed that demonstrate the basic, necessary features of this architecture.
This method is limited to valid topology results from the analysis software. The project scope excludes analysis setup and element threshold choice. Thus, the starting point of this method begins with a valid three-dimensional FEM optimal topology model.

The method will result in parametric geometry in the CAD model. While the execution of the method will only demonstrate the concept on the simplified cases, this thesis will lay out an architecture for this concept to be applied to more advanced features and shapes.

1.4 Anticipated Benefits

Allowing the engineer to evaluate factors such as manufacturing processes and parametric intent while relying on the computer to do the shape selection and fit would speed up this process 2x-4x. The results section contains the contributions of a simple implementation of this architecture. This simple implementation already shows substantially increases in time savings over manual topology model interpretation.

1.5 Conclusions/Recommendations

This method increases the speed in which designers can interpret topology optimization results. Future work should be done on each stage of the architecture to improve the capabilities and extend its versatility when used in conceptual and preliminary design studies.
2 Literature Review / Background

The following review of literature and background information is a collection of past and presently accepted engineering and computer science oriented literature associated with the proposed research into topology optimization and CAD geometry. Its purpose is to establish a background for the reader regarding what has been done previously in the areas of topology optimization, parametric CAD, and feature recognition. It also shows how this previous research will be built on and incorporated in this thesis. From the problems discussed in the above sections, the research requires an investigation of the following topics:

2.1 – Topology Optimization
2.2 – Shape Fitting/Recognition
2.3 – Feature Mathematics
2.4 – Advanced Parametrics
2.5 – Size Optimization
2.6 – API Programming

2.1 Topology Optimization

Though this thesis does not directly address specific aspects of topology optimization, background information pertaining to topology optimization will give the
reader a better comprehension of the types of topology models utilized and the researchers responsible for its development. Equally important is the understanding of the existence of voids and how they are utilized in this method.

As discussed in the delimitations, a commercial topology optimization engine will be chosen and used to accomplish the proposed integration. While the diverse techniques of topology optimization are not germane to this thesis, a firm understanding of this background research is invaluable to future work with the resulting topology models. Currently, researchers in the area of topology design and shape optimization have made significant progress, especially in the two major methods, homogenization [1] [2] [3] [4] [10] [11] [15] [17] [21] [25] and hard/soft kill [6] [19] [23] [24] [26] [27].

2.1.1 Homogenization

The homogenization method is a material distribution method. It is based on the use of composite materials with microscopic voids. Thus, the material is made up of two distinct components, solid material and empty voids. [1] This method has its roots in material microstructures and is used to evaluate the properties of materials with many infinitely small holes inside the domain of a uniform mesh of four-node iso-parametric elements. This is similar to the porosity and density of bones; the density is higher on high stressed areas and lower in low stress areas. Upon convergence, a contour plot of the densities reveals the optimal distribution of densities. This process accurately predicts the optimal topology in structural cases. However, according to Magalhaes, “its boundary is non-smooth and requires subsequent post-processing.” [20] Additional information about this method can be found in available literature. [3] [11]
2.1.2 Hard/Soft Kill Method

One of the more commonly used methods is the hard-kill method and was introduced by Xie and Steven in 1993. This method relies on the idea of changing material properties (i.e. Young’s Modulus) of the elements with less stress in the FEM to zero, essentially removing the material from the model. [27]

This method relies on two distinct parameters; the evolutionary rate and the removal rate. The removal rate (RR) determines the level of stress under which elements, if any, should be removed. This process is repeated for each convergence and the RR is increased by an increment called the evolutionary rate (ER). Once the stress level in all the design regions are below the objective stress level, the process is complete. The hard-kill method has some advantages over the homogenization method; it is simpler, able to automatically re-mesh, and has progressively decreasing computations for each cycle.

Similar to the hard-kill method, the soft-kill method, first developed by Walther and Matteck, uses a comparable approach aside from the concept of turning the elements on and off. [26] Instead the method uses a simple relationship on the stress levels in each element to vary its modulus. [6] Thus, each element gradually fades and is removed, giving the soft-kill method a slight advantage over its predecessor.

2.1.3 Manufacturing Constraints

Some methods of topology optimization include the ability to include some simple manufacturing constraints into the optimization process. These can include manufacturing concepts such as draw direction if the part needs to be pulled from a mold, from one or both sides. Other constraints can include extrusion paths, for cases where topology results follow a predefined path of the extrusion and the results are constant.
throughout the geometry along the path. Other examples include constraints on minimum member size and pattern repetition and grouping.

Manufacturing constraints often result in very different optimal topologies than those done without. It is assumed that the results of the unconstrained problem are global minima for the optimization problem, whereas those with manufacturing constraints are local minima.

2.1.4 Commercial Options

Discovering the most favorable topology of any specific design is a difficult problem for which there are a limited number of commercial options. When these commercial topology optimization programs are used in conjunction with pre/post processing programs they become significantly more useable. Some commercial options are available, including software from FE-Designs, MSC, ANSYS, and Altair Engineering.

OptiStruct from Altair Engineering is one example of a commercial topology optimization. Often it is used jointly with their software HyperMesh; which is used for both pre- and post- processing of the topology model. This software is specifically mentioned here because of its importance in developing this framework of topology to CAD. More information on OptiStruct can be found on the Altair website (http://www.altair.com).
2.1.5 Existence of Voids

Along with the concept of topology optimization, the concept of voids is introduced. According to the topology optimization problem, any result from the optimization process will result in a sub-domain of the original design space.

The void insertion follows the rules of Boolean algebra for regular sets. Regular sets are used in geometric modeling to ensure model validity and eliminate “dangling” edges or faces. They define geometrically closed sets that are of the following equation:

\[ A = k_i A \]  

(2-1)

This equation states that if the closure (k) of the interior (i) of a given set is equal to the same given set, then the set is regular. Operations involving regularized sets are differentiated from ordinary ones by a superscript * to the right of the operator. Figure 2-1 depicts an example of a non-regular set and how it fails to obey equation 2-1.

Figure 2-1 A non-regular set that fails to match the closure of the interior
Voids are essentially negative features used to represent absence of material. They can be represented mathematically as shown in equation 2-2.

\[ A - ^*B = ki(A - B) \]  \hspace{1cm} (2-2)

This equation states that with regularized sets, a set A minus a set B yields the closure of the interior of set A minus B. Showing this graphically yields Figure 2-2.

![Figure 2-2 Inserting void B into a set A and showing their regular set](image)

2.2 Shape Fitting/Recognition

Shape fitting algorithms exist in many different areas of research, most having little or no connection to topology optimization to CAD. However, the concepts applied in these research areas have a potentially large impact on the problem of topology to CAD. By the end of this section, the reader can have an understanding of past methods employed to recognize shapes or features and how they could be employed in this thesis.

The most common area in industry to find pattern or shape recognition methods is in the processing of point cloud data. Usually, these methods are employed for reverse engineering purposes with scan data. Some researchers have suggested that even in the realm of reverse engineering there is a need early on to address later stages of design.
These researchers also suggest that segregation of the data, coupled with the use of standard feature types such as extrusion and revolutions help in planning for processes such as manufacturing and other downstream processes, something a B-rep model does not do. [5]

Some researchers have used feature recognition in conjunction with computer-aided machining. These researches present the concept of recognizing features with “associated manipulation handles”. This is a similar problem in topology to CAD; both methods are trying to show that “some mechanism is needed to connect the shape representation to the parametric representation.” [16]

Recently, more sophisticated methods have been developed to import the topology as smooth models with advanced recognition. [18] [13] For example, Hsu et al. use a fully automated technique to create spline cross-sections and sweep a complete surface over them. They make some very important points throughout their work. First, the statement that they, “hope to transfer the topology optimization result into a smooth CAD model that can be used in later design stages,” shows that they fully intend to use this process as a springboard for further analyses and processes. Next, they state that, “any CAD software can be used to construct the three-dimensional CAD model,” showing that their method is intent on being software independent. Finally, one of the most important parts of the paper states that, ”too many unnecessary details will complicate the interpretation process, but not necessarily reduces the compliance of the structure”. This gives at least a partial basis for using simpler features to represent complicated voids in the FEM without significantly reducing the overall quality of the model. [14]
2.3 Feature Mathematics

There are various methods for controlling geometry in CAD. The purpose of this section is to give the reader one view of how geometric features in CAD can be parameterized. Understanding this manner of parameterization will give the reader an understanding of how the simple cases shown in this thesis can be extrapolated to more complex examples.

Often, in commercially available CAD packages, objects are represented using simple constructive solid geometry (CSG). [7] [8] This method of representing geometry has some benefits; usually in cases where simple geometry and high mathematical accuracy are desired. CSGs are used to define more complicated shapes, but require the use of one or more Boolean operations.

Along with CSGs, CAD-based geometry can be based on profiles and spine rules which define a feature. Examples of features include extrusions, revolutions and sweeps. Defining a feature requires a set of cross-sectional curves and a set of spine rules. Cross-sectional geometry loops are made of any number of curves. One method of defining all the curves in the cross-section uses rational Bezier curves of degree-n. A closed loop for the cross-section is required in most cases to define a solid object. Thus, mathematically a cross-section can be represented as shown in equation 2-3.

\[
B = \{ C_i(t): i = 1,2,3,... \} \\
\]

In equation 2-3, B is the set of all curves, C_i(t) that are joined end to end and enclose a single area (a closed loop), i.e. define a cross-section. Similarly, the rules of a
spine of a feature can be represented as an open or closed loop of curves \( D_j(u) \) that define a spine \( S \). This is shown algebraically in equation 2-4.

\[
S = \{ D_j(u) : j = 1, 2, 3, \ldots \} \tag{2-4}
\]

A rich variety of solid objects can be defined parametrically using any combination of cross-sections and spine rules. These parametric features can be described mathematically as shown in equation 2-5.

\[
O = R(\{B\}, \{S\}) \tag{2-5}
\]

In this equation, \( O \) is the solid feature, defined by sweeping the set of cross-sections \( B \) over the set of spine rules \( S \). It is desirable to define a feature this way due to the ease of control of the solid through parameterization. Each cross-section and spine rule can be completely constrained through the use of dimensional and geometric constraints. Thus, controlling the feature becomes merely an issue of defining the proper cross-sections and spine rules. Creating more complicated geometry in this manner does not require as many different shapes and operations as was required with CSGs.

For example, a common feature type in CAD is the extrusion. This shape has a single cross-section and a single linear spine rule. The cross-section of this feature is made of single first degree Bezier curves (straight lines). The lines are constrained as shown in Figure 2-3.
These features can be changed and updated by simply modifying the parameter values associated with the curves. For the purposes of this project, only the single planar cross-section with single normal spine rules (extruded features) will be examined. However, the concepts shown can be built upon for more complex shapes with complex parameter schemes. Examples of more complex features are shown in Figure 2-4.
2.4 Advanced Parametrics

Today, designs in modern CAD systems utilize various parametric schemes. Some schemes are intended for constraining various aspects of the part to certain design criteria. More complex schemes take into account the design criterion as well as allowing for multiple designs to be represented within a single model. Often this concept is used in cases where thousands of similar parts need to be produced from a single CAD design. More complex parametric schemes can be employed which allow for many...
seemingly dissimilar designs to exist within a single part-family. The purpose of this section of background is to give the reader an idea of the types of parametrics intended for parts and part-families that derive from a topology to CAD process and some of their concepts.

Leaders in industry are converting to strongly parametric CAD designs. The most important implementation of these designs involves the ability to control the model according to rules-based parameters. When the first commercial parameter-based CAD system appeared, most model parameters were driven geometrically; that is, parameter values mapped to dimensional values. Later another form of geometric relationships was developed in terms of CAD design. These are often referred to as geometric constraints; meaning geometric properties are established, such as parallelism, perpendicularity, etc. In the context of this thesis, only dimensional parameter-driven geometry is considered.

However, more robust parametrics began to be applied and engineering-based parameters (physics-based) began to be employed. For example, a long cantilevered rod has two main parameters, height and radius. In geometric based parameter schemes, the values of the parameters that map to the height and radius are the actual intended values. In an engineering based parameter scheme, the parameter mapping to the radius could be based on a table of available stock material. The parameter mapped to the height could then be based on the parameter for the radius, the stock material information and an equation for minimum deflection based on a known loading condition. This concept is shown in Figure 2-5.
One example of these enhanced parametrics is the concept of a part-family. In the past, part-families often only referred to simple parts such as screws and bolts that needed to be retrieved often from a part library. Due to the increased power of modern CAD packages, the same concept of modifying part parameter values to morph from one instantiation to another can be applied to much more complex parts. For example, some features can be suppressed and unsuppressed based on Boolean parameter values which allows for complex topology changes. Coupling this ability with robust geometry value changes and engineering rules-based parameter changes allows for completely new designs within a single part file. Figure 2-6 shows a simple case in which complex
topology features are suppressed by a Boolean parameter. Minor features like these are often suppressed to make design changes or before sending the geometric data off to an analysis program where mesh properties are important.

![Two fidelity levels of a part file controlled by a Boolean parameter](image)

Figure 2-6 Two fidelity levels of a part file controlled by a Boolean parameter

This type of part-family parameterization is used in many different areas of industry. A gas turbine engine designer could create high fidelity models of various parts of the engine, but reduce the fidelity for computationally expensive analyses simply by ‘turning off’ the unnecessary features. Similarly, the same part in the engine could be ‘reparameterized’ to become a completely new design without creating a new CAD model.

### 2.5 Size Optimization

Coupling shape optimization with size optimization is very powerful in reducing weight for critical parts of an assembly. The purpose of this section is similar to that of
the section on advanced parametrics; by the end, the reader should have an idea of how size optimization could be employed on a part derived from a topology to CAD operation to further reduce weight.

Various methods exist for performing size optimization. In the context of a CAD-centric approach, all of these methods involve controlling the parameters of the features from within the CAD model. The new model is meshed (preferably in an automated fashion) and the weight and analysis characteristic are measured. Depending on the size optimization method, design parameter changes and iteration count varies.

Most common is the standard approach of size optimization where the various geometric parameters are considered design parameters (input variables). The weight and analysis characteristic are considered to be analysis variables (output variable). The design parameters are updated according to an optimization scheme such as a reduced gradient, if the design space permits, or something more complicated such as a genetic algorithm. Generally, this will be the slowest, and can be the most accurate (though not guaranteed, depending on the optimization scheme and noise in the design space).

An improvement on this method is to include a mesh refinement strategy to increase the overall speed. Often in the beginning of a size optimization procedure, the accuracy of the scheme is lower than at the end of the procedure. Because of this, the mesh in the early stages of the optimization procedure is made much coarser. Once the procedure develops, the mesh is refined, especially in areas where accuracy is important.

A different approach to size optimization uses statistical methods for driving to an optimal value. Design of experiments (DOE) is used extensively in engineering to generate a response surface model of the design. A response surface model is generated
from a number of iterations based on the number of design variables and the accuracy of the model. The model can then be minimized according to the response surface. Often this process is repeated, centered about the assumed optimum, each iteration gaining a higher resolution near the global optimum. Compared to the direct method of size optimization described above, this method is often faster but less accurate depending on the quality of the response surfaces.

2.6 API Programming

The Application Programming Interface, or API, of a software program is a collection of routines and data types that give access to its core functionality. The APIs of several different software packages are leveraged to implement the presented method. Thus, this chapter is meant to give the reader a background in modern CAD/CAM/CAE APIs and their general functionalities as well as introduce the APIs required by this thesis.

2.6.1 Types of API Programs

Generally there are three types of APIs: macros, program-specific languages, and high-level programming language with program-specific functions. [12] Programs utilizing an API can usually be executed in different ways. First, they are called internally as a set of compiled library calls that are loaded into memory for use and then unloaded upon the completion of the commands. Next, they are a stand-alone executable that runs on the operating system outside of the software environment that can gain access to the data available. Finally, they are simple uncompiled scripts that are interpreted line by line, executing each command as it is read.
Scripts are computationally the slowest, but generally the fastest to create. Compiled internal programs are computationally the fastest, yet often the most difficult to program (often due to difficult compiling/linking). The main disadvantages of creating custom API programs are the upfront costs of developing the software and the initial learning curve to understand the API. [23] For use in this paper, two main API’s will be used; the Tcl (Tool Command Language) API inside of Altair’s HyperMesh and the Open C API in UGS (now Siemens) NX 4.0.

### 2.6.2 HyperMesh API

Tcl is a script based language that is simple to use and employs human readable syntax. Tcl as a programming language can be a stand alone-application (as long as the operating system has a Tcl interpreter) or can be imbedded in application programs (as with the case of the HyperMesh API). Often Tcl is coupled with Tk (Tool Kit) which is a powerful tool for creating GUIs (Graphical User Interfaces) quickly and easily. HyperMesh 8.0 come with Tcl/Tk v8.2.3 embedded within the software. This API gives the user the ability to leverage many useful tasks within HyperMesh, such as accessing nodal information, elemental information and values. Along with the API, HyperMesh provides a command history file for all actions performed interactively. Through this command file custom HyperMesh commands can be extracted and used directly within custom Tcl files.

Tcl comes with available standard input and output channels for passing data into and from other programs (i.e. C executables). This is done by calling the ‘exec’ command in Tcl and passing the C executable name with all input arguments through the standard output channel. The arguments are then ready for use in the C program. Once
the C program is finished and is ready to pass back information to Tcl, the information is parsed in through the standard input channel.

2.6.3 NX 4.0 API

UGS (now Siemens) NX 4 has a variety of different API’s available. For the context of this project, only the Open C API will be considered. The API is employed by including custom API libraries though standard header files for the various parts of libraries. Since the API is used directly within standard compiled C programs, this allows for both internal and external programs to be created. Generally, internal programs on Windows operating systems are created as dynamic-link library files (.dll) and are loaded into NX though a user command or a user-defined graphical user interface (GUI). External programs on the other hand do not require the software to be open, but rather perform their task on the specific part files directly by referencing the library commands. These types of programs generally work best in scenarios where the time to load NX is a bottle neck to the speed of the over all system, (i.e. long optimization procedures, design of experiments (DOE)).

2.7 Literature Review Summary

The material reviewed in this chapter is meant to give the reader a background on topics and literature that will be used or referenced in this thesis. The topology optimization review explained the process of arriving at a topology optimization model and introduces the existence of voids in the model.

The sections on shape fitting, recognition feature mathematics, and software APIs were presented to give the reader some background on some important concepts that will
be used in the method presented in this thesis. Shape fitting and recognition are presented to give the reader a sense of the areas where it has already been applied, such as point cloud analysis. Feature mathematics is presented to give a basis for portions of the method. The software APIs were presented to show how communication exists between various software environments.

Finally, advanced parametrics and size optimization were presented to give the reader a sense of the proposed end result of this method and things to consider when carrying out each step.
3 Method

This chapter explains an approach to creating the link between topology optimization results and CAD feature based geometry. The method explains the process of void selection and segregation and its importance in feature fitting. The method then goes through the process by which features are fit to voids. Finally, the method explains the process by which parametric feature data is transferred from the generic feature data to a specific CAD design feature through the use of an API.

Each step of the method: void selection/segregation, parametric feature fitting, and CAD feature updating, is explained in general terms according to the thesis objectives set forth in chapter 1. An actual implementation of the method is given in the next chapter and does so according to the thesis delimitations.

3.1 Void Selection and Segregation

As explained in Chapter 2, voids are created during the topology optimization process. Often, the voids are too complex for the designer to consider them as a single parametric feature. Thus, user interaction is required, to interpret the voids according to the intent of the designer. More specifically, the designer needs to define and limit the searchable area for the parametric feature according to their design intent. This input gives the feature some initial characteristics and location. The initial characteristics and
locations are simply that the shape is now constrained within a sub-set of the original void. The shape is limited in that it cannot assume configurations that would have fit the original space.

Segregating the searchable area of the shape introduces the idea of non-unique shape fitting, i.e. the same void can be interpreted differently by different designers. Figure 3-1 shows an example of a two-dimensional complex void that is interpreted differently due to the design intent of the engineer.

Figure 3-1 Two interpretations of a complex void based on designer intent.
In Figure 3-1, the first interpretation uses the entire void as searchable space for a parametric feature. This results in a more complex feature, which better characterizes the void. Often exact fitting is used in cases where parametrics and manufacturability are less of a concern. Alternatively, the search area in the second example segregates the void into two distinct features. This can be characterized by circumstances where matching the void is important, but due to the simplified features, parameterization and manufacturing are now easier to perform. Specifically, features are now easier to user for creating tool paths and the few interior corners are ready to be approximated with small radii. Also, the shape is now defined by a smaller set of parameters, rather than the possibly hundreds of vertices needed to define the volume of the void in the topology model.

The void selection and segregation does not define the geometry that characterizes the void, but rather the non-searchable areas. In the first case the entire void was given as searchable space for shape fitting. The shape can more fully characterize the void, but does so as a single complex shape. In the second case, the features are limited to the user-defined searchable areas and do not represent the void as well as the more complex shape. However, they better represent the design intent of the engineer by interpreting the void as two distinct interpreted features. As individual shapes, they can also be controlled according to design rules specified for each new feature.

Selecting the void search area is accomplished in various ways. The most basic method of selecting is to manually select the components of the FEM void (such as vertices, lines, faces) and indicate that the void is defined by the volume that they
enclose. Manual selection refers to selecting the components one at a time or using simple tools to select groups based on user input. Simple void segregation can be performed effectively with a manual process. Many pre/post processors used in analysis software where topology optimization is performed have built in selection tools. For example, some tools allow for nodal selections based on an arbitrary window drawn by the user. Figure 3-2 shows a group of vertices on a topology model void selected through the use of a selection tool.

![Figure 3-2 Manual void selection using a ‘by window’ selection tool](image)

In all cases of manual selection, the void segregation occurs as part of the void selection process, i.e. the selection of the boundary of the void is also what segregates the void in its various segments. This is shown graphically in Figure 3-3.
Note that in Figure 3-3 the first image shows the void being segregated from the side and encompassing the boundary intended for the void. However, it can be seen in the second image that the void is actually much more complex than what has been selected. According to the segregation process, the void is, due to the design intent of the engineer, limited to the void segregation zone. This bounding volume of the searchable space for the void is shown graphically in Figure 3-4.
More automated selection processes may also be employed to do void searches, but at some point the ability for a designer to limit the searchable space according to design intent is necessary. Though there are automated possibilities of limiting the searchable space of a void based on the void complexity, this thesis only addresses the manual selection process.

Selecting and segregating the void, according to design intent, results in geometric data. This data can be nodal data, line data or even surface information. Regardless of the geometric data type, the information is now ready to be fit with a parametric feature with sets of cross-sections and spine rules.

3.2 Fitting the Parametric Feature

A parametric feature is fit to the available space once the void data is selected and segregated. As discussed in Chapter 2, the features used in this thesis are based on sets of cross-sections with spines to define the blending rules between them. This section of the method focuses on the concept of fitting the cross-sections and spine rules based on the available defined by the void selection and segregation.

3.2.1 Cross-section Information

The cross-sectional geometry in a two-dimensional model can be defined by a set of first degree Bezier curves. From this set of Bezier curves, the void is best represented by an n-sided figure matching the geometric data from the void. As the purpose of this method is to represent the voids with less complex and more controllable features, a simpler shape with few sides is desired. Thus there are two competing objectives in
fitting a cross-sectional shape: (1) limit the shape complexity and (2) limit the error from approximating the void with a simpler shape. Figure 3-5 shows a graphical representation of these two objectives and the Pareto front that is created.

More complex shapes can be achieved by weighting the approximation error more than the feature complexity. The inverse is true for achieving simpler features. This allows the designer to take the segregated voids interpreted in step one and fit a feature of desired complexity and fit based on a weighting ratio. Figure 3-6 shows the same segregated voids shown in the second half of Figure 3-1, but now with different weights on the minimization objectives.
The first example in Figure 3-6 shows the shape fitting scheme with more importance given to minimizing the approximation error. The second example shows the same searchable void regions now fit with shapes with more weight given to the minimization of the number of geometric parameters.

One issue that can arise from fitting simpler shapes to complex voids is the high unexpected loss of fidelity of the original shape. The voids generated in topology optimization models are considered to represent the global optimum in terms of the overall minimum weight or volume. The shape fitting procedures presented here fit the new features based on geometric characteristics. This can have unexpected results if the
design space is such that small deviations in the geometric design result in large deviations from the analytical optimum. This sensitivity to change is reflected in the steepness of the graph in Figure 3-7.

This problem can be addressed in two ways. First, this method is intended to emulate an already existing process currently used in industry. The deviations from the optimum are already occurring in the industrial use of this process and have so far not been a significant drawback to its importance. Second, some of the loss from the design change can be regained through the use of size optimization on the new geometry. This is important to consider as size optimization fits the shape according to analysis characteristics, which can be much different than the geometric fit. On the left in Figure 3-8 is an example of geometry fit to geometric characteristics, and on the right is different geometry fit according to an analytical optimization process, i.e. size optimization.
3.2.2 Creating a Volume Based on Spine and Cross-section Information

Due to the equal importance of the cross-sectional and spine information on the shape of a parametric feature, the fitting of the two feature components is considered to be coupled. The method for interpreting the spine rules needs to be an iterative process, re-fitting the existing geometry each time cross-sectional information is added. In the two-dimensional examples shown in the previous section, the rules for the spine are implied to be a single, straight line, normal to the plane of the cross-section.

Often the voids encountered in optimal topology models are not constant along their primary axis. Figure 3-9 shows an example of a cross-sectional view of a void in a topology model. The figure shows an example of the three-dimensional void being interpreted with a single (horizontal) cross-section, oriented in one of the primary coordinate directions of the model and a single (vertical) spine oriented normal to the cross-section. The method starts here at the simplest case and fits the cross-section according the average cross-section along the spine.
If the shape does not represent the void the shape needs to iterate and become more complex. The new iteration is then evaluated. Figure 3-10 shows an example of an iteration process with new spine directions and cross-sections. This iteration process can be completed depending on the complexity level specified by the designer.

3.3 Creating a CAD Model Link

Once the dimensional values and locations associated with the newly fitted shapes are discovered, they are sent to the CAD model via the CAD system API. The CAD API
routines for geometry creation are specific to each CAD system and have various levels of programming complexity. However, all of the creation routines for geometry require on a minimum level, the raw data produced from the shape fitting process or a derivation of it.

If the parametric feature only requires a single cross-sectional shape with a linear spine then the command for an extrusion (often also referred to as protrusion or pad) is sufficient and controls the feature as expected. If the feature requires more cross-sections, or a more complicated set of spine rules, then often a sweep feature or blended feature is sufficient. In cases where there are both complicated spine rules and multiple cross-sections, a mesh surface feature (sometimes referred to as a multi-section loft or thru-curves mesh) is required.
4 Implementation

The previous chapter was designed to give the reader an outline on how this process of topology to CAD could be accomplished for any parametric feature fitting to void geometry. This chapter will focus on an implementation of this method, using the delimitations outlined in the first chapter. This chapter will also walk through the same steps as Chapter 3 and show an implementation for each step.

The implementation demonstrated here uses (UGS) NX 4.0 and (Altair) HyperMesh 8.0. The implementation is intended only to show that this process could be carried out and not as a full blown implementation, as many of the steps are done in a rudimentary fashion and not as they would be done in an industrial setting. Instead, by observing a simple implementation, the reader should see the effectiveness of the method.

The topology optimization models that are used from within HyperMesh exist as FEMs. The original geometry that was used to create the topology model derives from an NX 4.0 part file. The void selections and segregations are performed in HyperMesh, using the HyperMesh Tcl API. The shape fitting algorithm is done independently inside of a C executable file. Finally, the update of the CAD model is accomplished through a stand-alone executable that accesses the Open C API inside of NX.
4.1 Void Selection and Segregation

In order to segregate the search space, the user is employed to do the void segregation. Once the user has located a void and decided on the void segregation, the user begins the process by selecting the bounding nodes of the void. In order to leverage the internal nodal selection tools, specifically the ‘by window’ selection tool, the user needs to orient the model so that the void spine direction is normal to the screen. This orientation allows for the screen transformation matrix to be used for shape-fitting purposes.

Once the user has oriented the model correctly, the user selects the nodal domain using the built-in tools in HyperMesh nodal selection tools. Although the user can select single nodes individually, time is saved by selecting groups of nodes using the ‘by window’ option. The nodes are collected through the HyperMesh Tcl routine:

```tcl
*createlistpanel nodes 1 "Select Void Points"
set point_cloud [hm_getlist nodes 1]
set num_pts [llength $point_cloud]
set i 0
foreach nodeId $point_cloud {
    set nodex [hm_getentityvalue NODES $nodeId "x" 0 ]
    set nodey [hm_getentityvalue NODES $nodeId "y" 0 ]
    set nodez [hm_getentityvalue NODES $nodeId "z" 0 ]
    set ptstring "$ptstring $nodex $nodey $nodez"
    set pts($i) $nodeId
    incr i
}
```

Through this snippet of code, all of the nodes, both those selected individually and those selected by other means, are stored into a string as x, y, and z coordinates. To simplify the implementation, when the node list contains only three nodes, the shape is assumed to have three vertices. There is no need to send the nodes off to the shape fitting algorithm. This also applies to four sided figures as well. Obviously, this concept
implies that the optimization of fitting the shape is left to the designer’s interpretation.

Figure 4-1 By window selection method for selecting nodes in HyperMesh

Figure 4-2 By window selection results
Figure 4-3 shows an example of selected nodes that segregates the intended void from the other voids in the model. Issues with this implementation of the void selection and segregation involve the ability of the user to select all the nodes and the trade off between time to select the nodes and the accuracy of the selection. If you select the nodes using the window option, you may select too many nodes (meaning those nodes not included as part of the intended void). If you select nodes using the single node option it may take an extensive amount of time to get all the nodes necessary. One option is to use the window selection technique and then use the single node selection to add in missing nodes or deselect those nodes not intended as part of the search area is an option. This implementation heavily relies on the manual selection process and does not implement a more automated approach for nodal selection. Because the implementation
utilizes nodal coordinates for shape fitting, it disregards the other FEM components that can be used, such as faces or lines.

At the end of the nodal selection and segregation, the final output is a string of X, Y, and Z coordinates for all of the selected nodes of the void. These nodal values along with the screen transformation matrix is stored in memory and prepared to be used in the next step.

4.2 Fitting the Parametric Feature

Once the spine information and nodal coordinates are ready from the previous step, this information is passed into the command line arguments of a custom C program which transforms the nodal points using the transformation matrix and transforms the points into a new coordinate system where the z value can be set to zero, effectively projecting the points onto a single plane.

Since there are more than four selected nodes, we will seek for the best-fit circle (recall that a three- or four-sided figure is chosen automatically, only when there are three or four nodes selected).

Upon transforming the points, a least squares operation is performed to rate the goodness of the fit of an arbitrary circular shape (remember, three and four sided shapes are assumed based on nodal locations).
The optimization algorithm then varies the x, y, and r values and the circles are re-evaluated until a best fit is found. In this implementation an exhaustive search approach is employed due to its ease of use and implementation. Once the shape has been best fit to the nodal cloud, only the values of x, y, and r are passed back to the

$$\text{Min}_{C_x, C_y, C_r} \left\{ S(C_x, C_y, C_r) = \sum_{i=0}^{n} \left[ \left( \sqrt{(C_x - P_{xi})^2 + (C_y - P_{yi})^2} - C_r \right) \right]^2 \right\}$$

s.t.
- $C_{x_{\text{min}}} \leq C_x \leq C_{x_{\text{max}}}$
- $C_{y_{\text{min}}} \leq C_y \leq C_{y_{\text{max}}}$
- $C_{r_{\text{min}}} \leq C_r \leq C_{r_{\text{max}}}$

Figure 4-4 Least squares algorithm for fitting a circle to the cloud of points.
HyperMesh API script. From there the script creates a HyperMesh instantiation of the recognized geometry. This gives the user a direct understanding of what is occurring with each void feature without having to switch to the NX model for verification. Concurrently, the values and types of the feature are written out to a text file to await the user approval of the visual feature in HyperMesh. Upon the termination of the final feature and approval, the user then releases the text file of shapes to the CAD model for updating.

The parametric feature is fit simply for a circular cross-sectional extruded feature. The direction of the feature is based upon the viewing matrix that was used when the designer did point selection simple to simplify the process.

This implementation of fitting a parametric feature has issues in a few different ways. The least squares method is the fastest and easiest method to implement. However, some of the drawbacks with least squares include being susceptible to groupings of points that can ‘pull’ the resulting circle. Least-squares is also not a very robust method for fitting shapes more complex than a circle. More complex and robust methods such as minimizing area of the Boolean intersectional area or even the volume difference between the intended feature and the original void could be better.

This implementation of fitting a parametric feature demonstrates that already-available routines can be used to fit cross-sectional geometry and spine rules according the available data from the selection and segregation step.
4.3 Creating the Link to the CAD Model

The link back to the original CAD model is done by taking the nominal values of the feature information and passing that into an API program in NX. The values from the fitting routine are passed back into the HyperMesh script and from there stored into a text file to be released to the CAD model as a real feature. Before it is released, the feature is created locally in the HyperMesh model using geometry features to give immediate feedback on how the shape fitting fared and the location and size of the feature.

This implementation of the link works well as it does exactly as it is intended to do. The feature information is collected (regardless of its quality) and translated into the CAD native feature set. Below are some Open C routines for translating the raw data into CAD features.

```c
// Add lines to a sketch
tag_t createLines(tag_t sketch, double *pt1, double *pt2, int blank){
tag_t line[1] = {NULL_TAG};
    UF_CURVE_line_t line_data;
    line_data.start_point[0] = pt1[0];
    line_data.start_point[1] = pt1[1];
    line_data.start_point[2] = pt1[2];
    line_data.end_point[0] = pt2[0];
    line_data.end_point[1] = pt2[1];
    line_data.end_point[2] = pt2[2];
    int err = UF_ERROR(UF_CURVE_create_line(&line_data,&line[0]));
    if(err != 0){
        return NULL_TAG;
    }
    UF_ERROR(UF_SKET_add_objects ( sketch, 1, line ));
    return line[0];
}
tag_t createExtrude(uf_list_p_t loop_list, uf_list_p_t features){
    double ref_pt[3];
    char *taper_angle = "0.0";
    char *limits[2] = {"-10.0", "10.0"};
    int english_units = 2;
    UF_FEATURE_SIGN create = UF_NULLSIGN;
    UF_ERROR(UF_MODL_create_extruded(
        loop_list,
taper_angle,
```
limits,
ref_pt,
direction,
create,
&features));
}

This implementation lacks some functionality that is not crucial but still would be desirable in this process. As the feature is intended to be fully-parametric in the CAD system, it would be desirable to implement a dimensioning scheme using the values of the cross-sectional geometry and the spine rules.

Once the text file from the last step is released, the information for each shape is parsed into a custom NX Open C API executable, called directly from the HyperMesh API. The executable takes the values of each shape along with the direction vector of the spine and creates extruded features with circular cross-sections as Boolean operations in the CAD file. Each feature is removed and then the part file is saved, reflecting the changes.
5 Results

This section examines the implementation of the proposed architecture on a set of case studies. Each case study has its own set of possible manufacturing constraints. The results from these case studies demonstrate the process of the implementation and examine some of the specifics successes and issues associated with each. The purpose of these results is to show the effectiveness of the method through the use of the simple implementation.

5.1 Simple Torsion Bar

The first case study examined is a simple cantilevered bar with a torque about the primary axis. The analysis model is shown below in Figure 5-1. The model is fully constrained on one end and has a torque applied on the other end, centered about the primary axis. The first case from this torsion bar example uses manufacturing constraints on the topology optimization model. The second case uses no manufacturing constraints and results in a very different optimized topology model. Both cases show the original topology followed by the newly recognized geometry and some of the successes and issues in arriving at the final geometry.
5.1.1 Torsion Bar with Manufacturing Constraints

The simple torsion loaded cantilevered beam is optimized first, using a split draw direction constraint, as described in Chapter 2. The resulting topology is shown below in Figure 5-2. Notice that due to the draw constraint, the resulting topology is primarily in the specified draw direction. Because of this, the various voids in the resulting topology are first interpreted in the primary draw direction.
In this case, there are ten total voids interpreted in that direction. Figure 5-3 gives a breakdown of the voids in that direction. The voids will all be interpreted individually. The intent is to represent each void as a single extruded shape with a constant cross-section. However, as will be shown below, not all of the shapes are able to be recognized as a single shape due to some limitations in the implementation.
The first void interpreted (the void labeled #1), shown in Figure 5-4 can be interpreted differently depending on the fit of the shape and the intent of the engineer. As explained in the implementation, the shape can be interpreted as a circle, three-sided or four-sided figure. The most complex figure is the four sided and has a fairly average fit. The three-sided figure decreases the complexity of the shape at the cost of a worse fit. Finally, fitting the void with a circular cross-section provides a good fit and the lowest level of complexity.

![Figure 5-4 Single void interpreted differently by more complex shapes](image)

This same process of complexity vs. fit is applied in a similar manner to the rest of the 10 voids in the model. Some voids could not be fit within reason with any of the available shapes. This limitation is due to the implementation only allowing for these simple shapes. For example, Figure 5-6 shows a void more complex than can be fit with the available shapes.
The most likely method to fit this void is to implement a more complex polygonal shape. As none are available in this implementation, one work around is to fit the void with more than one simple shape (void segregation). Figure 5-6 shows the void being fit with three overlapping circles.
The void is segregated according to user interpretation. A similar process is employed for the shaped labeled 10 in Figure 5-3. In this case, void segregation could simply be avoided through a five-sided figure. The geometry, once the void recognition from the top view is completed, is shown below in Figure 5-7.

Once all the voids are recognized in the main draw direction, the voids viewed from the side are also recognized. These eight voids are shown in Figure 5-8.
All of the voids in this direction are interpreted with the simple figures and the resulting geometry is shown below in Figure 5-9.

Once the voids have been interpreted in the first two coordinate directions, the resulting approximation geometry is finished and is shown in Figure 5-10 and is overlaid on the original topology in Figure 5-11.
Figure 5-10 Final resulting geometry model

Figure 5-11 Resulting geometry compared with original optimal topology model
5.1.2 No manufacturing constraint

The torsion bar optimization without any manufacturing constraint reveals a much different topology and is shown below in Figure 5-12. The topology basically consists of an internal void with the shape of a square tube, extruded along the primary axis of the bar. On the exterior, there are ridges that run the length of the model.

![Figure 5-12 Torsion bar optimal topology without manufacturing constraints](image)

The void along the primary axis can be interpreted differently. Both the square cross-section and circular cross-section have error in the fit. This is due to the shape of the internal void being more like a square shape with rounded corners.
The edges of the model were recognized along the same axis are the primary interior void. The final view of the recognized geometry along the primary axis of the beam is shown below Figure 5-14.
Just as before in the previous case, the model is viewed from the side and two voids are interpreted. These voids are shown below in Figure 5-15.

Figure 5-15 Recognized geometry viewed from the side

The resulting solid that approximates the original topology is now complete and is shown below in Figure 5-16 and is overlaid on the original topology in Figure 5-17.

Figure 5-16 Final approximating geometry after all voids have been interpreted
The interpretation of this topology model also included some issues that could not be resolved with the current implementation. The implementation assumes through holes for each of the voids. The main internal void as seen from the front view is not a through hole, but rather is rounded on the far end. This drawback can be avoided by implementing the ability to use multiple cross-sections and control the depth of the feature based on void characteristics.

Table 5-1: Volumes for torsion bar topology results and approximations

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Design Space</th>
<th>Optimal Topology</th>
<th>% of Design Space</th>
<th>Approximate Topology</th>
<th>% of Design Space</th>
<th>Diff. (as % of D.S.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torsion Bar – no constraints</td>
<td>40000</td>
<td>9320</td>
<td>23.3%</td>
<td>10912</td>
<td>27.3%</td>
<td>+4.0%</td>
</tr>
<tr>
<td>Torsion Bar – split draw</td>
<td>40000</td>
<td>10427</td>
<td>26.1%</td>
<td>13053</td>
<td>32.6%</td>
<td>+6.5%</td>
</tr>
</tbody>
</table>

Table 5-1 shows a comparison of the optimal topology volume compared to the newly created approximating geometry. The percentage of the optimal topology and
approximating geometry are shown compared to the original design space. Both cases of
the torsion bar (with and without manufacturing constraints) show a slight increase in
volume when approximated with simpler parametric features.

5.2 Cantilevered Cube

The next case study examined is a cantilevered cube with a distributed load on the
top surface. The cube is constrained on one side only at the top and bottom edges. The
cube is first examined without any manufacturing constrains, and later with a split draw
direction constraint. The cube analysis model is shown below Figure 5-18.

![Cantilevered cube problem with a distributed load](image-url)
5.2.1 Cube with Manufacturing Constraints

The cube optimization with a draw manufacturing constraint yields the topology shown below in Figure 5-19.

Due to the manufacturing constraint the resulting topology is very close to being an extruded shape, with a single cross-section. The voids are interpreted in the same manner as explained in the previous section. Due to the simplicity of the shape of the final results, only the voids seen from the side view are interpreted. These shapes can be seen in Figure 5-20 and overlaid on the original topology in Figure 5-21.
Figure 5-20 Final recognized geometry for the cube with manufacturing constraints

Figure 5-21 Final interpreted geometry overlaid on the original topology model
5.2.2 Cantilevered Cube with no Manufacturing constraints

The results from the constrained cube problem with no manufacturing constraints are shown in Figure 5-22.

![Figure 5-22 Cube topology optimization results with no manufacturing constraints](image)

Notice that this is assumed to be the global optimum for the topology optimization process and is a much more complex shape than the previous example. Also note that because the topology is more complicated, any approximations to the topology with simple shapes will have higher error in the approximation and will contain more shapes to create the model. The voids are selected and segregated from three different views, shown below in Figure 5-23.
Due to the high complexity of the topology in this case, the comparison of the interpreted shape and the original topology shows much more error than the previous examples. Some of the error can be alleviated by implementing more complex parametric features into the method. However, much of the error is due to the fact that the void segregation is much more difficult to perform on this model manually. More automated methods could perform better due to their ability to segregate voids based on geometry characteristics and not user ability.
Table 5-2: Volumes for cube topology results and approximations

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Design Space</th>
<th>Optimal Topology</th>
<th>% of Design Space</th>
<th>Approximate Topology</th>
<th>% of Design Space</th>
<th>Diff. (as % of D.S.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cantilevered Cube – no constraints</td>
<td>1000</td>
<td>223.2</td>
<td>22.3%</td>
<td>377.0</td>
<td>37.7%</td>
<td>+15.4%</td>
</tr>
<tr>
<td>Cantilevered Cube – split draw</td>
<td>1000</td>
<td>332.7</td>
<td>33.3%</td>
<td>393.1</td>
<td>39.3%</td>
<td>+6.0%</td>
</tr>
</tbody>
</table>

Table 5-2 shows a similar comparison of the optimal topology volume and the approximating geometry as seen in the previous section. Once again when compared to the original design space, both cases show an increase in volume when approximated with simpler parametric features. As noted above, the case with no manufacturing constraints has a higher increase in volume as compared to the other volumetric changes.
6 Conclusions

The objective of this research was to investigate the possibility of generating CAD parametric features based on topology optimization results. Specifically, an implementation was developed that performs the basic tasks of the method. The implementation creates geometry based on geometric characteristics of the topology model. This implementation was performed on a set of test cases and was able to demonstrate the following operations:

- Void selection through the use of native software selection tools
- Void segregation through user interpretation of the topology model
- Shape fitting of simple circular shapes based on a least squares algorithm
- Shape fitting of simple three and four-sided shapes based on user input
- Creation of CAD geometry through the use of the UGS NX 4.0 Open C API

The implementation for generating CAD parametric features is not yet able to perform void selection and segregation using automated methods based on the topology model’s geometric characteristics. The implementation is also not yet able to create parametric features based on more than one cross-section or with complex spine rules or both. These additions to the implementation require more research into the methods of evaluating the void characteristics, possibly from more than just the single iso-surfaces that the current topology models derive from.
The implementation for the topology to CAD architecture is written in the HyperMesh Tcl API, stand-alone C code and in the Open C API for NX 4.0. To implement this method on different systems and software packages, the appropriate code changes are necessary as well as translating the existing implementation to different software packages.

Results from the implementation were done on two different models with two different sets of manufacturing constraints. The effectiveness of the implementation for these test cases was based on comparisons between the optimal topology information and the final interpreted geometry. Conclusions were drawn on the implementation and method based on successes and issues that arose during the testing of the test cases.

It was found that many of the topology optimization results can be readily used to generate CAD parametric features. Furthermore, the ease of interpretation is heavily based on the ability of the user to segregate the void based on design intent and on the manufacturing constraints placed on the model. Many of the voids in optimized topology models that utilize manufacturing constraints can be approximated with polygonal shapes that use simple linear spine rules. The more complex shapes are difficult to approximate but this is directly related to the difficulty of manufacturing and parameterization. This research successfully established an architecture by which topology optimization results can be used to generate CAD feature based geometry. The method presented created an architecture that eliminates much of the work of new feature selection and fitting from the topology results. It gives the designer the flexibility to take into consideration engineering criteria and feature complexity based on parameterization needs and manufacturability.
6.1 Future Work

As mentioned often throughout this thesis, a large amount of research is available in the area of automating void segregation and selection. These continued efforts of automation can be classified as improvements on the first step of the topology to CAD method. Improvements on the current implementation of the method would include development of more complex shapes and shape fitting algorithms. These methods could be used to fit shapes to much more complex topology and work directly with new methods of void segregation.

Specific ways to improve upon the current method include, but are not limited to, the following:

- Carefully studying and quantifying the shape fitting methods presented here.
- Incorporating size optimization into the shape fitting method to combine geometric shape fitting with analytic shape fitting.
- Utilizing geometric constraints, such as parallelism, perpendicularity, etc.
- Creating shape fitting procedures based on manufacturing constraints used during topology optimization process.
- Direct parametric modeling of smooth topology IGES results should also be investigated.
7 References


