2014-07-01

A Computational Hybrid Method for Self-Intersection Free Offsetting of CAD Geometry

Garrett Clark Bodily
Brigham Young University - Provo

A Computational Hybrid Method for Self-Intersection Free Offsetting of CAD Geometry

Garrett Bodily

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

C. Greg Jensen, Chair
Christopher A. Mattson
David T. Fullwood

Department of Mechanical Engineering

Brigham Young University

July 2014

ABSTRACT

A Computational Hybrid Method for Self-Intersection Free
Offsetting of CAD Geometry

Garrett Bodily
Department of Mechanical Engineering, BYU
Master of Science

Surface offsetting is a valuable tool used in Computer Aided Design (CAD). An offset surface is a collection of points that are at a constant distance from another surface. An offset surface is created in CAD by selecting a surface and then specifying the distance that the surface is to be offset. If a surface is selected and a distance of D is specified, then the resulting offset surface should always be distance D from the original surface.

The surface offset tool can be used for many applications. Modeling of composites or other layered manufacturing processes rely heavily on offset surfaces. Thin walled parts such as injection molded components are often modeled using the offset tool. Coating processes can also be modeled using the offset tool.

Modern CAD systems have surface offsetting tools and are widely used throughout industry. However, CAD systems often fail to produce valid results. The process of surface offsetting can often result in surface self-intersections as well as surface degeneracies. Self-intersections and degeneracies make the surfaces invalid because they are physically impossible to create and CAD systems cannot use these invalid surfaces to represent solid bodies. The surface offset tool is therefore, one of the most challenging CAD tools to implement. The process of avoiding, detecting and removing surface self-intersections is extremely challenging. Much research in the field of CAD is dedicated to the detection and removal of surface self-intersections. However, the methods proposed in the literature all suffer from robustness problems.

The purpose of this research is to introduce a method that creates valid offset surfaces and does not suffer from the problem of creating surface self-intersections. This method uses a numerical approach that approximates the offset surface and avoids all self-intersections. Because no self-intersections are created, the method does not require intersection tests of any kind. The value of this method is demonstrated by comparing its results with results from leading CAD systems.

Keywords: offset surface, self-intersection

ACKNOWLEDGEMENTS

I would like to thank my committee chair Dr. Jensen for helping this thesis become a reality. I would also like to thank my dear wife, Merilee, for supporting me and taking such good care of me while I pursued this goal.

TABLE OF CONTENTS

LIST OF TABLES ..... **vii**

LIST OF FIGURES ..... **viii**

1 Introduction **1**

1.1 Problem Statement 1

1.2 Thesis Objectives 2

1.3 Research Delimitations 3

1.4 Document Organization 3

2 Background **5**

2.1 CAD Basics 5

2.1.1 BREP Topology Structure 5

2.1.2 Geometric Data 6

2.1.3 Parametric Geometric Data 7

2.1.4 Bezier and NURBS 7

2.1.5 Derivatives of NURBS 8

2.2 Surface Offsetting 8

2.2.1 Offset Surface Definition 9

2.2.2 Self-Intersections 10

2.3 Tessellated Surfaces 13

2.4 Computational Methods 13

2.4.1 Computational Offset Surfaces 14

2.4.2 Volumetric Offsets and Distance Fields 15

2.4.3 Surface Contouring 16

2.4.4 Surface Reconstruction 17

2.5 Pros and Cons of Offset Methods 17
3 Method 19
3.1 Introduction 19
3.2 Geometric Method 19
3.3 Computational Method 21
3.4 Hybrid Method 22
3.5 Original Research 23
3.6 Convert Parametric CAD Geometry to Tessellated Geometry 24
3.7 Create Tessellated Offset Surface 24
3.8 Convert Offset Back to CAD 25
3.8.1 Project Geometry Onto Offset 25
3.8.2 Recreation of Basic Parametric CAD Geometry 26
3.8.3 Recreation of Free-Form Parametric CAD Geometry 26
3.9 Conclusion 27
4 Implementation 29
4.1 Implementation in 2D 29
4.1.1 Conversion of 2D Parametric CAD Geometry to Tessellated Geometry 30
4.1.2 Signed Distance Field Creation 31
4.1.3 Marching Triangles 32
4.1.4 Geometry Projection 34
4.1.5 Line Segment Splitting 34
4.1.6 Recreation of Curves 36
4.1.7 Final Product 38
4.2 Implementation in 3D 39
4.2.1 Conversion of 3D Parametric CAD Geometry to Tessellated Geometry 39

4.2.2 Tessellation Storage in Half-Edge Data Structure 39
4.2.3 Signed Distance Field Creation 41
4.2.4 Marching Tetrahedron 42
4.2.5 Geometry Projection 43
4.2.6 Face Subdivision by Modified Ear Clipping 45
4.2.7 Recreation of Vertices 47
4.2.8 Recreation of Edges 48
4.2.9 Recreation of Faces 49
4.3 Putting it All Together 54
4.4 Accuracy of the Hybrid Method 54
5 Results 57
5.1 Test Parts 57
5.2 Commercial CAD Systems 59
5.2.1 NX Results 59
5.2.2 Pro Engineer Results 63
5.2.3 CATIA Results 67
5.2.4 SolidWorks Results 71
5.2.5 Inventor Results 75
5.3 Summary of CAD System Results 79
5.4 Offset Tool Results 80
5.5 Summary of Hybrid Method Results 84
5.6 Test Case 7 85
5.7 Conclusion 88
6 Conclusions 89
REFERENCES 91

LIST OF TABLES

Table 4-1: Surface Offset Methods 49

Table 5-1: Accuracy of CAD Results 80

Table 5-2: Summary of All Results 85

LIST OF FIGURES

Figure 2-1: Local Self-Intersection 11
Figure 2-2: Global Self-Intersection 11
Figure 3-1: CAD Offset Method 20
Figure 3-2: Computational Method 21
Figure 3-3: Hybrid Method 23
Figure 4-1: Curve Tessellation 30
Figure 4-2: Signed Distance Field 32
Figure 4-3: Marching Triangles 33
Figure 4-4: Contouring 34
Figure 4-5: Geometry Projection 35
Figure 4-6: Tessellation to CAD 38
Figure 4-7: Surface Tessellation 39
Figure 4-8: Half-Edge Data Structure 41
Figure 4-9: Marching Tetrahedron 42
Figure 4-10: Geometry Projection 43
Figure 4-11: Modified Ear Clipping 46
Figure 4-12: Vertex Creation 48
Figure 4-13: Parameterization of Tessellated Surface 50
Figure 4-14: Surface Genus 51
Figure 4-15: Control Point Grid 52
Figure 4-16: Extrapolating Point Locations 53
Figure 4-17: Tessellated Surface to NURBS 54
Figure 4-18: Error Introduced by Contouring 55

Figure 5-1: Simple Test Parts 58
Figure 5-2: Complex Test Parts 58
Figure 5-3: NX Part 1 60
Figure 5-4: NX Part 2 60
Figure 5-5: NX Part 3 61
Figure 5-6: NX Part 4 61
Figure 5-7: NX Part 5 62
Figure 5-8: NX Part 6 62
Figure 5-9: NX Part 7 63
Figure 5-10: Pro/E Part 1 64
Figure 5-11: Pro/E Part 2 64
Figure 5-12: Pro/E Part 3 65
Figure 5-13: Pre/E Part 4 65
Figure 5-14: Pro/E Part 5 66
Figure 5-15: Pro/E Part 6 66
Figure 5-16: Pro/E Part 7 67
Figure 5-17: CATIA Part 1 68
Figure 5-18: CATIA Part 2 68
Figure 5-19: CATIA Part 3 69
Figure 5-20: CATIA Part 4 69
Figure 5-21: CATIA Part 5 70
Figure 5-22: CATIA Part 6 70
Figure 5-23: CATIA Part 7 71
Figure 5-24: SolidWorks Part 1 72
Figure 5-25: SolidWorks Part 2 72

Figure 5-26: SolidWorks Part 3 73
Figure 5-27: SolidWorks Part 4 73
Figure 5-28: SolidWorks Part 5 74
Figure 5-29: SolidWorks Part 6 74
Figure 5-30: SolidWorks Part 7 75
Figure 5-31: Inventor Part 1....76
Figure 5-32: Inventor Part 2....76
Figure 5-33: Inventor Part 3....77
Figure 5-34: Inventor Part 4....77
Figure 5-35: Inventor Part 5....78
Figure 5-36: Inventor Part 6....78
Figure 5-37: Inventor Part 7....79
Figure 5-38: Hybrid Method Part 1 81
Figure 5-39: Hybrid Method Part 2 81
Figure 5-40: Hybrid Method Part 3 82
Figure 5-41: Hybrid Method Part 4 82
Figure 5-42: Hybrid Method Part 5 83
Figure 5-43: Hybrid Method Part 6 83
Figure 5-44: Hybrid Method Part 7 84
Figure 5-45: Hybrid Method Part 7 86
Figure 5-46: Genus 1 Surface 86
Figure 5-47: Cutting the Surface....87
Figure 5-48: Cutting the Surface....87
Figure 5-49: Test Part 7 Modified 88

1 INTRODUCTION

In the last few decades, the engineering industry has been revolutionized by the advent of computers and more specifically by Computer Aided-Design (CAD). CAD systems have transitioned from simple 2D drafting programs to fully 3D solid modeling environments in which a designer can use a variety of tools to quickly and accurately design products. Sketched based tools, Boolean operations, advanced surfacing techniques and many other powerful tools allow the designer to create virtually anything. One of the tools that can be used in the design process is an offset surface. Surface offsetting is a valuable tool in the disciplines of engineering and manufacturing. It is also, however, one of the most difficult and problematic geometric operations to perform. State-of-the-art CAD systems that are used by even the most advanced engineering companies often fail to produce offset surfaces.

1.1 Problem Statement

Offset surfaces are required for many engineering and manufacturing steps and applications within the design process. State-of-the-art commercial CAD systems can produce offset surfaces of single sheets or of solid bodies. These CAD systems, however, often fail to produce the offset of a complex object once the offset distance has passed a critical value. This lack of robustness for offset generation among commercial CAD systems greatly limits their capabilities and can often lead to extensive “work-arounds” and secondary solutions that result in lost productivity and lower quality CAD models.

1.2 **Thesis Objectives**

The goal of this research is to demonstrate a new method of creating offset surfaces that is general enough that it can create a valid offset surface for any geometry at any distance. The author has yet to find a CAD system with offset capabilities robust enough to do this, and such a tool would be of great value to engineers and product designers. In order for this method to successfully overcome the challenges presented by offset surfaces, several key issues will need to be addressed. Self-intersections of individual surfaces will need to be detected or avoided, global intersections will also need to be detected or avoided, and a method for trimming and sewing the offset surfaces together will also need to be presented.

In the process of researching this topic, methods have been found that address these different issues. Each of the existing methods solves a very specific problem of the offset process but none of them are considered to be a general solution. These known methods, which will be discussed in greater detail in the following chapter, are the groundwork of this research. The objective of this thesis is to show that by combining known methods and creating a few new methods a powerful and robust general solution can be created.

The objectives of this research are:

1. Develop a robust general method for error free surface offsetting.
2. Use this general method to show how it addresses the different problems that are inherent to surface offsetting.
3. Demonstrate the value of this method by creating offsets for several parts that cannot be offset by leading CAD systems.

1.3 Research Delimitations

In order to limit the scope of this project, only a small set of geometric elements will be used to demonstrate the usefulness of the tool. CAD systems can create many different types of curves and surfaces, but for this project the geometric entities will be limited to the following.

Curves: Arcs, Lines, NURBS curves

Surfaces: Planes, Cylinders and NURBS surfaces

This may seem like a very small sample of the geometric entities but this list represents both the easiest elements to offset and also the most difficult to offset. By proving the method on the easiest and also the most difficult it can be assumed that all other geometric elements can be handled. The research will also be limited in scope by implementing the tool in only one CAD system. The tool will be created in such a way that it will only interact with the CAD system to extract the input geometry and then to display the result but all calculations in between are independent of the CAD system. This tool could in theory be easily ported to any other CAD system.

1.4 Document Organization

In the following chapters this thesis will first discuss what methods have already been created and the benefits and drawbacks to each one. Next the idealized method will be discussed. Chapter four will cover the actual implementation of this method. Chapter five will demonstrate that the implemented method works and has value. Chapter six will discuss what conclusions can be drawn from the results of this research and will also suggest what future work is needed.

2 BACKGROUND

2.1 CAD Basics

In order to better understand the process of offsetting and the complexities it involves, it is valuable to first understand the inner workings of a modern CAD system. In order for a CAD system, or a computer in general, to be able to represent and perform operations on geometry, it must be represented in a numerical format. Points, curves and surfaces must be stored in way that they can be represented as a set of numbers and the relationship between the various geometric entities must also be stored. For the purposes of this research both 2D and 3D shapes are considered.

2.1.1 BREP Topology Structure

Modern CAD systems rely on a data structure known as a Boundary Representation or BREP (Braid 1975). The BREP model is a topological structure that stores the geometric information of a solid model in an organized way that is best suited for CAD. A solid object in CAD is represented by its boundary that bounds a closed volume. This boundary, or shell, is a collection of faces (Braid 1975). Each face is a region of a surface that is defined by a geometric surface (e.g. plane, sphere, cylinder, etc.) and a set of edges known as a loop (Braid 1975) (Forsyth 1995). A loop is a closed set of one or more edges that bound a region of the surface. Outside of this region the surface is not used. A face must have at least one loop but can have

more than one if the face contains holes. Edges are portions of curves (e.g. line, arc, etc.) that are bounded by vertices and vertices are just points in space (Braid 1975).

Topology does not define the actual shape of the solid object; it only defines how the various pieces of the object are connected. The actual shape of the object is defined by the various geometric entities associated with the topology. Faces are associated with surfaces, edges are associated with curves and vertices are associated with points.

2.1.2 Geometric Data

These various geometric entities must be stored by the CAD system in a mathematical form that is easy for the CAD system to work with. The easiest piece of geometry to store is the point which is just a vector of two values (x, y coordinates) in 2D and three values (x, y, z coordinates) in 3D. Curves are more complex to store and surfaces are more complex still. For curves and surfaces there are various ways to mathematically represent them, two common forms of curve and surface equations are the explicit and the implicit forms. The explicit form takes the form of $y = f(x)$ in 2D and $z = f(x, y)$ in 3D and the implicit form is $f(x, y) = 0$ in 2D and $f(x, y, z) = 0$ in 3D. These two forms are known as non-parametric because the individual coordinates depend on each other (Piegl 1997) (Rogers 2001) (Shah 1995) (Faux 1984). These forms of equations have many uses are commonly used in mathematical fields but in the field of CAD there are several limitations that make them sub-optimal. Explicit and implicit forms are coordinate system dependent which makes it difficult to apply geometric transformations (e.g. rotating) (Rogers 2001) (Shah 1995). Multi-valued functions cannot be represented easily and even simple shapes such as a circle would need to be broken into multiple functions in order to capture the entire curve (Faux 1984) (Rogers 2001). It is also difficult to compute consecutive points on a curve in order to render it on a computer, because of the relationship between the

coordinates in explicit and implicit functions point locations may be difficult to solve for or may even be un solvable (Shah 1995) (Faux 1984).

2.1.3 Parametric Geometric Data

Equations that are better suited for CAD systems are known as parametric equations (Faux 1984) (Piegl 1997). Parametric equations separate each coordinate to depend on an independent variable known as a parameter (Shah 1995) (Piegl 1997) (Faux 1984) (Rogers 2001). The range of these parameters is typically normalized to 0 to 1 but is not required to be so (Shah 1995) (Rogers 2001) (Piegl 1997). The parameter range can be scaled to any value without changing the shape of the curve or surface, therefore the parameterization and the equation for a curve or surface in parametric form is not unique (Rogers 2001) (Piegl 1997). Only one parameter is required for a curve but two parameters are required for a surface (Piegl 1997) (Rogers 2001) (Piegl 1987). Unlike the explicit and implicit equations, geometric transformations can easily be applied to parametric equations (Faux 1984) and it is easy to compute points in order along a curve or surface which makes it easier for a CAD system to display the geometry (Shah 1995) (Faux 1984).

2.1.4 Bezier and NURBS

A very useful parametric curve and surface equation that is used heavily in CAD is the Bezier curve and also the non-uniform rational b-spline, or NURBS, curve. The Bezier curve was developed by Pierre Bezier and also Paul De Casteljau who were both looking for a way for designers to easily define a freeform shape in engineering drawings (Shah 1995). NURBS were first investigated by Ken Versprille (Piegl 1987) and are a more flexible version of a Bezier curve. A NURBS curve as well as a NURBS surface can be made to exactly represent a Bezier

curve and Bezier Surface respectively (Piegl 1987) (Coquillart 1987), therefore, CAD systems mainly support NURBS curves and are a vital part of CAD systems (Farin 1983) (Floater 1991) (Rogers 2001) (Piegl 1987) (Shah 1995). NURBS are more flexible than Bezier curves because they are not constrained to a specific number of control points but can have any number of control points (Shah 1995), and despite having any number of control points are easily evaluated (De Boor 1972) particularly for low degrees, the most common degree found in CAD being degree three.

Part of the reason that NURBS are so vital to CAD is that not only can they represent any free from shape, they can also represent common shapes exactly. NURBS curves can be made to represent lines, arcs, circles, and conic sections (Coquillart 1987) (Piegl 1987) (Shah 1995) (Rogers 2001). Also, NURBS surfaces can be made to represent common surface types such as planes, cylinders, spheres and cones (Rogers 2001).

2.1.5 Derivatives of NURBS

NURBS curves and surfaces can also be derived like other curve and surface equations. These derivatives are important because from them, key properties of the curve or surface can be determined. Properties that are valuable to know for a NURBS curve or surface are the tangent vectors, normal vectors and curvature.

2.2 Surface Offsetting

Offset surfaces are an important geometric operation and have many applications. Offset surfaces can be used for Rapid Prototyping (Liu 2009) (Kumar 2002) (Shen 2010), Tolerance Analysis (Liu 2009), CNC Path generation (Liu 2009) (Pottman 1995) (Forsyth 1995)(Piegl 1999)(Kumar 2002) (Maekawa 1999)(Shen 2010) (Rossignac 1986) (Zhang 2011)(Faux

1984)(Seong 2005), Shelled models (Liu 2009) (Forsyth 1995) (Maekawa 1999), Filleting (Liu 2009) (Rossignac 1986), Thick Plates/sheet metal (Pottman 1995)(Faux 1984), Mold Creation (Forsyth 1995), Clearance/tolerance checking (Forsyth 1995) (Maekawa 1999) (Shen 2010) (Rossignac 1986)(Pavic 2008) (Zhang 2011)(Shah 1995), Is useful in both 2D and 3D (Coquillart 1987), Modeling of composites (Kumar 2002) (Zhang 2011), Robot path planning (Kumar 2002) (Maekawa 1999) (Shen 2010) (Rossignac 1986) (Zhang 2011)(Seong 2005), Coating processes (Rossignac 1986). Mathematically it is a very simple geometric operation but in reality is very complex. Offsetting of 2D shapes is very well understood and many CAD/CAM systems can offset 2D profiles robustly but it is valuable to examine 2D offsetting because it offers a comparison between established methods and the method presented in this paper. Also, it is much easier to demonstrate methods in 2D.

2.2.1 Offset Surface Definition

Offset surfaces, also referred to in the literature as parallel surfaces (Maekawa 1999)(Forsyth 1995), are collections of points at a fixed distance from an object. These points may be inside or outside of the object (Liu 2011). The offset surface is also a special case of the Minkowski sum (Rossignac 1986) (Liu 2011) (Pavic 2008) (Varadhan 2004). The mathematical definition of the offset surface is very simple, however, it has proven to be a very difficult and complex operation (Liu 2011)(Tiller 1984).

$$\boldsymbol{O(t) = C(t) + \delta * N(t)} \quad \textbf{(2-1)}$$

$$\boldsymbol{O(u,v) = S(u,v) + \delta * N(u,v)} \quad \textbf{(2-2)}$$

The offset surface is itself a parametric equation, O is the offset surface, S is the original surface, d is the offset distance and N is the normal vector of the surface (Faux 1984) (Piegl 1999) (Pottman 1995) (Tiller 1984) (Farin 1989) (Seong 2005) (Elber 1991) (Kumar 2002)

(Piegl 1997) (Forsyth 1995). This equation is very simple and for many surfaces is trivial. For common curve and surface types such as lines, arcs, planes, spheres, cylinders and also a very special type of curve known as a Pythagorean Hodograph the offset curve or surface can be offset and represented exactly (Farin 1989) (Tiller 1984) (Kumar 2002) (Elber 1991) (Maekawa 1999)(Shen 2010). For freeform curves and surfaces that are represented by NURBS curves or surfaces, the offset curve or surface, in general, cannot be represented exactly or in the same form as the original shape and instead must be approximated (Piegl 1999) (Seong 2005) (Pottman 1995) (Tiller 1984) (Farin 1989) (Forsyth 1995) (Elber 1991) (Kulczycka 2002) (Jang 2005) (Piegl 1997). A large portion of the research with regards to offset curves and surfaces of NURBS revolves around approximating the offset within a specified tolerance while using the lowest number of control points and knots to do so.

The offset operation can be applied to single curves or surfaces or to entire 3D models (Liu 2011). For offsetting a solid body, each individual face must be offset and then the individual offset surfaces must be trimmed against each other in order to reconstruct a closed volume (Yoo 2009) (Forsyth 1995) (Liu 2011). As the surfaces are trimmed against each other changes in the topology can occur (Liu 2011). This process of trimming the offset surfaces together can be difficult when surfaces meet at very low angles; however, this problem is small compared to the problem of self-intersections.

2.2.2 Self-Intersections

Another area where much research is focused is the area of self-intersections. Self-intersections are the reason that even though the offset equation is very simple, creating robust offset surfaces is extremely complex. A self-intersection is when a curve, or surface, folds back on itself creating a physically impossible shape, this behavior is undesirable (Seong 2005)(Tiller

1984). There are two types of self-intersections discussed in the literature, these are local and global. Local self-intersections occur when the curvature of a surface is less than the offset distance (Forsyth 1995) (Seong 2005) (Tiller 1984) (Elber 1991) (Maekawa 1999) (Kumar 2002) (Faux 1984) (Pekerman 2008). Even in smooth and relatively simple curve or surfaces self-intersections can occur when the offset distance has passed the critical value defined by the curvature (Rossignac 1986). Figure 2.1 shows a simple curve that has been offset at two different values; the larger offset has passed the maximum curvature causing it to intersect itself. Global self-intersections can occur when two different points on a curve or surface offset to the same location even with relatively low curvature (Seong 2005) (Elber 1991) (Maekawa 1999). Figure 2.2 shows a curve that globally self-intersects, which occurs even though the offset distance has not passed the maximum curvature.

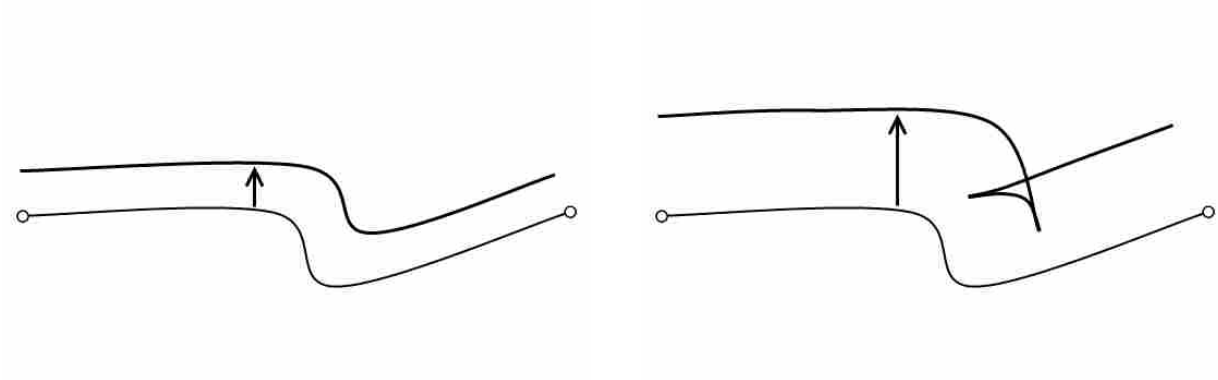


Figure 2-1: Local Self-Intersection

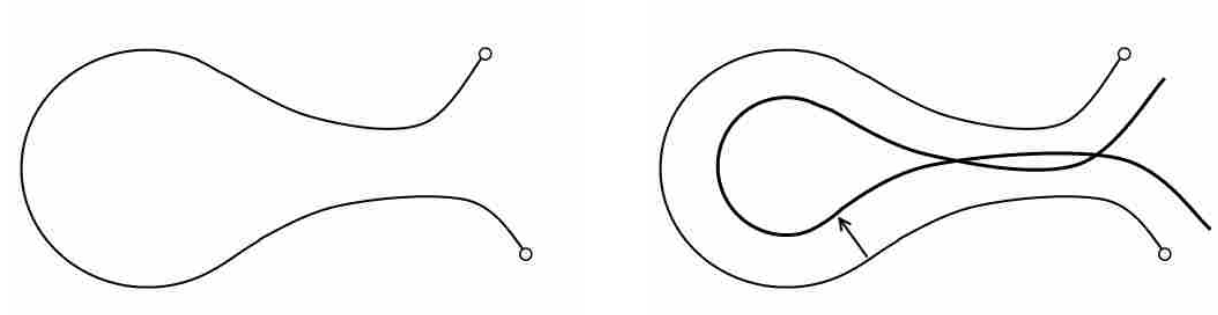


Figure 2-2: Global Self-Intersection

Self-intersections are difficult to locate (Elber 1991) (Thomassen 2001) (Elber 2009) and even more difficult to avoid or remove (Coquillart 1987) (Seong 2005). Various methods have been documented for locating and removing self-intersections from free-form curves and surfaces. In the case of planar curves, locating and trimming self-intersections has become very well understood and many robust algorithms have been developed. However, surfaces are a different story. It is much more difficult to find/trim self-intersections in surfaces than curves (Seong 2005) (Piegl 1999) (Pekerman 2008) (Thomassen 2001) (Jang 2005). In general methods that have been developed for detecting self-intersections in curves do not, or are difficult to, apply to surfaces. Algebraic methods have been developed for determining parameter values for the surface where S(u0, v0) = S(u1, v1) and u0 != u1 and/or v0 != v1 (Maekawa 1999) (Pekerman 2008) (Elber 2009) (Thomassen 2001) (Kumar 2002). This process is computationally expensive to locate a single point of the self-intersection. Rarely do self-intersections result in a single point but are in general typically a curve that must be discovered through numerical marching techniques (Seong 2005) (Elber 1991) (Thomassen 2001). Another method proposed in literature is a distance map between the original surface and the offset surface. The offset surface should be at a constant distance from the original surface but if self-intersections have occurred this distance can be less than the offset distance. This method is fairly robust and like the previous mentioned method can detect both local and global self-intersections (Seong 2005). Not all methods can detect both types of intersections (Elber 1991). By checking the direction of the tangent vectors in the original surface and the offset surface and checking for vectors that have reversed directions local intersections can be detected but not global intersections (Pekerman 2008). A method is also presented to locate self-intersections in

surfaces by looking for self-intersections of the iso-curves of the surface (Kumar 2002). This method is in general limited to very few surfaces. Most surfaces that intersect themselves do not do so in such a clean and easily treatable manner.

If any self-intersections are detected, there is still additional work that must be done in order to create a desirable offset surface. These self-intersections must be trimmed away leaving only the valid region of the offset (Elber 1991)(Forsyth 1995)(Kumar 2002). This step of the offsetting process can also be difficult for CAD systems to handle.

2.3 **Tessellated Surfaces**

In contrast to the parametric curves and surfaces used in CAD to represent geometry, a method for approximating geometry is a tessellated representation. Tessellated surfaces are collections of points and faces, where each face is a triangle. Triangle meshes are the most common method for approximating shapes (Jung 2004)(Jones 2006). Due to advancements in computing power and memory, tessellated surfaces can be very highly accurate (Flutter 2001) (Treece 1999). Tessellated surfaces are used widely in CAM, computer graphics, rapid prototyping, robotics, reverse engineering and FEA (Flutter 2001) (Yoo 2009) (Jung 2004).

2.4 **Computational Methods**

Computational methods are the process of taking very complex mathematical operations and approximating them with much simpler mathematical operations. These simpler operations must be performed many times over so the tradeoff is simpler operations but more of them. A large portion of the research surrounding computational methods focuses on finding ways or shortcuts to reduce memory usage (Pavic 2008) and also to increase the speed (Wang 2013).

However, the focus of this research is not on optimizing the computational routines but instead focuses on the robustness of the method.

2.4.1 Computational Offset Surfaces

Due to the widespread use of tessellated surface approximations, computational methods for calculating offset surfaces have evolved. Some methods are focused only on rendering the offset surface but not actually calculating the tessellated body (Dziegielewski 2010). Methods for calculating the tessellated offset have also been researched and involve moving the vertices of the original tessellated surface along the normals of the surface (Malosio 2009) (Pavic 2008). The normals can be calculated at the vertices by averaging the normals of the faces that connect to the vertex (KIM 2004). These methods do not avoid self-intersections but because the resulting surface is triangulated, more robust methods for finding and removing self-intersections have been created. The self-intersections can be removed from 2D slices of the offset (KIM 2004) or the individual faces of the offset surface can be intersected and trimmed using triangle-triangle intersection methods (Jung 2004) (Moeller 1997). However, even triangle-triangle intersections tests, although more easily computed than NURBS self-intersections, can become unstable in areas of cusps. Other methods for calculating offset approximations include a method in which each vertex is offset to a solid sphere, each edge is offset to a cylinder and each face is offset to a solid prism and all of these solids are united together to create the offset solid (Pavic 2008), fitting offset surfaces to scan data or point clouds (Liu 2009)(Zhang 2011), and also "Shrink-wrapping" the object (Overveld 2003).

2.4.2 Volumetric Offsets and Distance Fields

The most robust method for creating a tessellated version of the offset surface is a volumetric approach. In the volumetric approach, self-intersections are trivial (Jang 2005) (Shen 2010) (Liu 2011). In order to calculate an offset surface via the volumetric approach, a distance field first needs to be computed. A distance field is a grid of points in which at each point the minimum distance to the object is known, this distance can also be signed with negative generally indicating the point is inside the object and positive as the point is outside the object (Frisken 2006) (Yin 2011) (Jang 2005) (Jones 2006) (Varahan 2003). A distance field is essentially an approximation of the implicit function of the object (Frisken2006). As with many computational methods the smaller the spacing in the grid the higher the accuracy is (Shen 2010). In order to achieve higher accuracy and still maintain high speeds and low memory lots of research regarding distance fields is focused on developing methods for calculating the distance field quickly (Yin 2011) and also methods for storage and higher resolution (Pavic 2008) (Jones 2006) (Yin 2011).

Distance fields are used in many applications other than just offset surface calculations. Distance fields have found use in digital design (Frisken2006) (Yin 2011), boolean operations (Frisken2006) (Jones 2006), collision detection (Yin 2011) (Jones 2006), visualization (Yin 2011)(Jones 2006), round/fillet calculations (Frisken2006), medical imaging (Frisken2006), fluid simulations (Frisken2006) (Jones 2006), robotics (Frisken2006), and FEA mesh generation (Jones 2006).

Distance fields offer several advantages for creating offset surfaces. As mentioned before volumetric approaches (i.e. distance fields) make handling self-intersections trivial (Jang 2005)

(Shen 2010) (Liu 2011). Furthermore, changes to topology are also handled robustly and multiple offset surfaces can be generated from a single distance field (Frisken2006).

2.4.3 Surface Contouring

Once a distance field has been defined for a particular object, a method is needed in order to extract the offset surface. It is desirable that the extracted surface be closed and self-intersection free (Ju 2006). This process of extracting a surface from a distance field is known in the literature as contouring (Treece 1999) (Ju 2006) (Varahan 2003). The most well-known method of contouring is the marching cubes algorithm (Lorensen 1987). The marching cube algorithm considers a cell comprised of 8 vertices and each vertex may be either inside the offset surface or outside. This leads to a total of 256 (2^8 = 256) possible configurations for any given cube. These 256 possible cases can then be combined by symmetry into only 15 unique cases (Lorensen 1987) (Treece 1999) (Yoo 2009). However, some flaws have been found in the marching cubes algorithm and other contouring methods have been developed to address these flaws (Chernyaev 1995) (Treece 1999) (Varahan 2004). One weakness of marching cubes algorithm is the difficulty in capturing sharp features (Varahan 2003) (Ju 2006). A proposed solution to the sharp feature problem is the algorithm Dual Contouring that introduces the use of hermite data to recreate sharp features (Ju 2002). Another weakness of marching cubes is the occurrence of ambiguous cases where a surface may pass through a cube in multiple ways. This ambiguity can leave holes in the surface which is undesirable. Variations of marching cubes have been proposed to address this weakness (Varahan 2003) (Varahan 2004) (Yoo 2009) (Treece 1999) (Ju 2006) (Chernyaev 1995).

2.4.4 Surface Reconstruction

Of all of the methods for creating tessellated offset surfaces only one paper addressed the idea of recreating parameterized geometry. The method presented slices the triangulated offset surface using parallel planes and fits splines to the slices. These slices are then used to create skinned surfaces that fit to the offset surface (Jang 2005).

2.5 Pros and Cons of Offset Methods

Of the various offset methods presented, the two main areas are offsetting the individual faces of the solid and attempting to locate and trim away self-intersections, and approximating the offset using a distance field and a tessellated surface. The first method has the benefit of producing actual CAD geometry which is desirable because the geometry can be used in further modeling operations in the CAD system. The biggest draw back to the first method is that dealing with self-intersections is extremely complex and CAD systems have very limited capabilities to do so. The second method presents the exact opposite pros and cons. The volumetric approach can handle self-intersections, both locally and globally, as well as topological changes without issue but only produce tessellated surfaces. Tessellated surfaces are valuable for visualization and some engineering applications such as CAM and FEA but cannot be used to perform additional modeling operations in the CAD system. The purpose of this research is to draw on both areas of research and create a hybrid method that has both the advantages of dealing with self-intersections and topology changes robustly but also produces parameterized CAD geometry.

3 METHOD

3.1 Introduction

This chapter describes the steps that are taken to produce a self-intersection free offset surface. As mentioned in the background section, there are two main offset methods that are covered in the literature. These two methods are the geometric method and the computational method. In order to understand the hybrid method proposed by this research it is best to review the geometric and computational methods.

3.2 Geometric Method

Figure 3.1 shows the overall geometric method. This method begins with parametric CAD geometry and offsets each geometric element that comprises the shape (curves for 2D shapes and surfaces for 3D shapes). Equations 2.1 and 2.2 are used to generate the offset for each element. This equation does not avoid self-intersections. Therefore, depending of the complexity of the geometry self-intersections can occur. These self-intersections must be located and eliminated. Once the offset of each element has been produced, they are then trimmed against each other to create the offset shape. The trimming portion of the process can also be difficult and cause the process to fail.

The benefit of this method is that the final output is parameterized CAD geometry. This is advantageous because it allows designers and engineers to work with parametric CAD geometry and allowing further modeling steps to be taken in the CAD system. However, this method has the disadvantage of having to locate and eliminate self-intersections in parametric curves and surfaces. As mentioned in the previous chapter, locating self-intersections in 2D profiles is well understood and CAD systems have this capability, but locating self-intersections in 3D surfaces is extremely challenging and often prevents the CAD system from producing an offset surface.

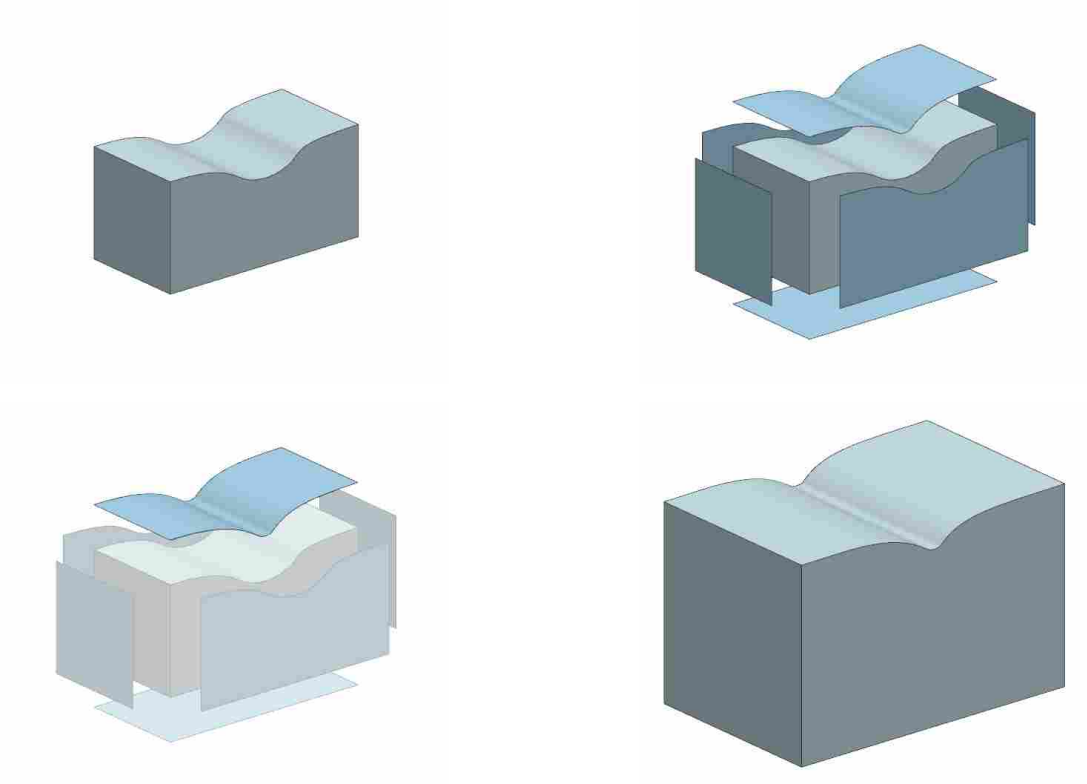


Figure 3-1: CAD Offset Method

3.3 **Computational Method**

The computational approach is described in figure 3.2. This method begins with a tessellated version of the geometry. From this tessellated model a signed distance field is created. The signed distance field is a grid of points where at each point the minimum distance to the object is known along with whether the point is inside or outside of the object. Once the signed distance field is created a process known as contouring is used to extract a tessellated surface that approximates the offset surface.

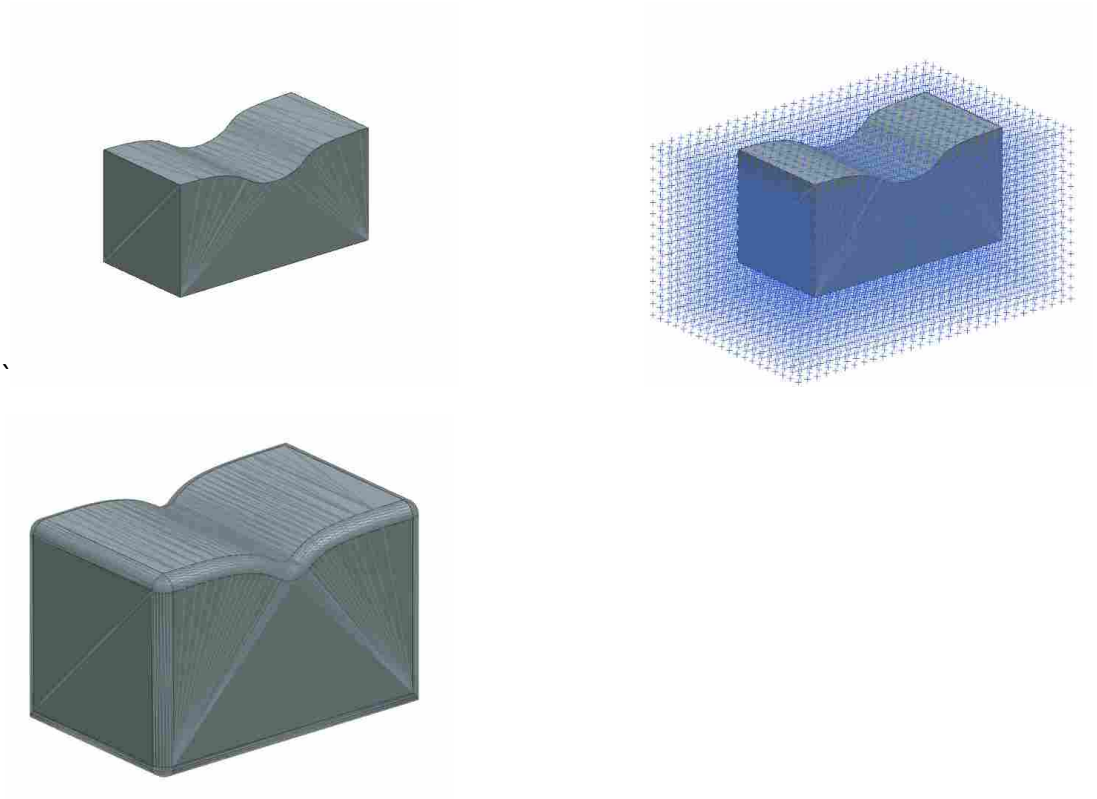


Figure 3-2: Computational Method

Unlike the geometric method, the computational method does not run the risk of creating self-intersections. Using a signed distance field with contouring guarantees that no self-

intersections will occur. Errors arising from trimming are avoided by the computational method. The disadvantage to this method is that the output a tessellated surface which is only an approximation of the offset and has no actual geometric entities associated with it. This disadvantage greatly limits the usefulness of the computational method because it cannot be used for further geometric modeling operations in the CAD system.

3.4 **Hybrid Method**

This research combines both the geometric and computational methods together to create a more robust general method. This hybrid method is illustrated in figure 3.3. In order to make this method fit into the work flow of CAD users both the input and output must be parametric CAD geometry. The first step, therefore, of the hybrid method is to convert the parametric CAD geometry into tessellated geometry. With this approximated form the computational method can then be used to create a tessellated offset surface. This approximated version now is no longer in the work flow of the CAD user therefore it must be converted back to parametric CAD geometry. The last step of the hybrid method is converting tessellated geometry back to parametric CAD geometry that CAD systems can use.

The hybrid method can be summarized as follows:

Step 1 – Convert parametric CAD geometry to tessellated geometry.

Step 2 – Create tessellated offset using computational approach.

Step 3 – Convert tessellated offset back into parametric CAD geometry.

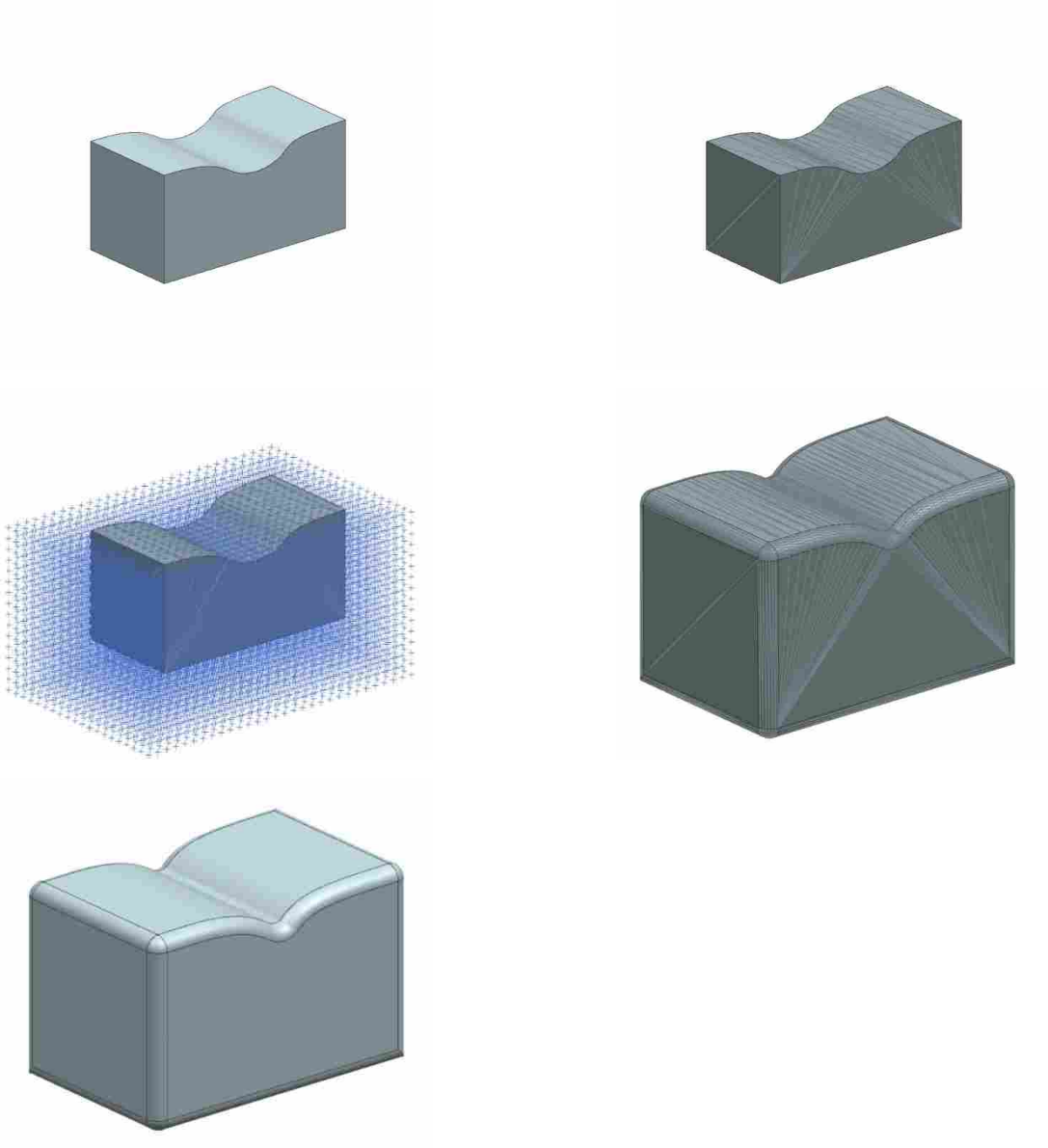


Figure 3-3: Hybrid Method

3.5 **Original Research**

In the above summery of the hybrid method, steps 1 and 2 are carried out by established methods described in the literature. However, step 3 is where the true value of this research lies.

The process of converting a tessellated offset surface back to parametric CAD geometry is the most challenging and important step of this process.

3.6 Convert Parametric CAD Geometry to Tessellated Geometry

CAD systems rely on the ability to convert geometric entities to simple representations in order to render the objects to the screen. Curves are approximated with line segments and surfaces are approximated with triangles. Modern graphics hardware on computers can draw lines and triangles very efficiently. Therefore, any part that is displayed by a CAD system is a demonstration of tessellated geometry. This method relies on the built in functionality of CAD systems to tessellate the geometry.

3.7 Create Tessellated Offset Surface

The computational method is now followed to create a tessellated offset surface. As described earlier, a signed distance field is a regular grid of points that surround the object. At each point the minimum distance to the object is known. It is also known if the point lies inside or outside of the object. Grid points that are located inside the object have a negative sign for their distance. Once a distance field is created the tessellated offset surface can be created using contouring. This method relies on a variation of the Marching Cubes algorithm known as Marching Tetrahedron. Marching tetrahedron groups the grid points into sets of four thereby forming tetrahedron shaped cells. Each cell is analyzed individually and based on the distances measured at each grid point the cell can be classified as one of three possible cases.

It is important to point out that the Marching tetrahedral algorithm is guaranteed to produce a self-intersection free offset surface. This is due to the fact that each cell is disjoint

from one another and the triangular faces generated by the algorithm are always contained within the cells.

3.8 Convert Offset Back to CAD

The final step of the hybrid method is to convert the tessellated offset surface back to parametric CAD geometry. This is the most difficult and most important step of the process. Methods do exist in the literature for converting tessellated geometry to parametric CAD geometry but these typically involve reverse engineering and tessellated models that do not have CAD models associated with them. Because the tessellated offset generated in this method has a CAD model associated with it, the geometric entities of the original shape can be used to help in the process of converting back to CAD.

3.8.1 Project Geometry Onto Offset

First, every point on the tessellated offset surface is projected back to the original shape. Or in other words, the nearest point is found on the original shape that corresponds to each point on the offset. The nearest point is used to determine which points on the offset belong to which geometric entity of the original shape. Once all points on the tessellated offset surface are associated with a geometric entity of the original shape the elements of the tessellated offset are analyzed for changes in geometry.

In 2D the tessellation is comprised of line segments where each segment is defined by two nodes. If the two nodes have different geometry then the line segment is split. In 3D the tessellation is comprised of triangular faces that are defined by three nodes. If the nodes do not all share the same geometry then the face is subdivided.

3.8.2 Recreation of Basic Parametric CAD Geometry

Once every element of the tessellated offset has been analyzed and split accordingly, the elements are grouped together based on the geometric entity they are associated with. For geometric entities that are simple geometry (e.g. lines, arcs, planes, cylinders, spheres, etc.) the offset can be calculated exactly and that portion of the tessellated offset surface can be replaced with the exact geometric representation.

3.8.3 Recreation of Free-Form Parametric CAD Geometry

As was mentioned in the chapter 2, offsets of free-form geometry must be approximated. Therefore, portions of the tessellated offset surface that correspond to free-form geometry must be fit with a NURBS curve or surface.

In 2D the line segments of the tessellated offset are replaced with NURBS curves that are fit to tessellated data. This process involves placing equally spaced points along the line segments and then using these points as control points to create a NURBS curve.

In 3D this process is more complex. Now the tessellated offset must be fit with a NURBS surface. In order to fit a NURBS surface to a region of the tessellated offset surface, the region must first be parameterized, or in other words, represented in 2D. The parameterization is accomplished with a method known as Least Squares Conformal Maps (Levy 2002).

Once the surface has a 2D representation, a regular grid of points is placed on the surface. For grid points that lie inside the boundary of the offset their 3D position is determined by placing the point on the tessellated face.

For grid points that lie outside of the tessellated region their 3D position must extrapolated using a technique that extends the grid lines using the points that lie inside the region as references. This method approximates where the points outside of the surface should be

positioned. The exact position of the points outside of the boundary is not critical because that portion of the surface will be trimmed away, but a regular grid of points is needed for a NURBS surface to be created. Once all faces have been fit with a surface and trimmed by their bounding curves they are combined together in the CAD system to create a solid body and the offset is complete.

3.9 **Conclusion**

As can be seen by the overall process, by combining various aspects of existing offsetting methods, a general hybrid method can be created. This method has the potential to become general enough and robust enough that any geometric object or any parametric CAD model could be offset to any distance, regardless of its complexity. The main contribution of this research is the process of using the original geometry to convert the tessellated offset surface back to parametric CAD geometry. This will allow the process to be inserted into the work flow of CAD users and allow them to create offsets that were not previously possible in CAD.

4 IMPLEMENTATION

In order to test the value of the proposed method it is necessary to create a working prototype of the software. This chapter details how the proposed method was implemented. The implementation was created using C++ in Microsoft Visual Studio 10, an open source library known as the Computational Geometry Algorithms Library (CGAL) and the CAD system NX 6.0. Two versions of the software were created and tested, the first applies the method to 2D shapes and the second applies the method to 3D shapes.

4.1 Implementation in 2D

As mentioned in previous chapters, CAD systems are capable of offsetting 2D shapes. Self-intersections and trimming in 2D is handled robustly by modern CAD systems. Therefore, it may seem unnecessary to implement the method in 2D. It is worth discussing the 2D version of the software strictly for demonstration purposes. The method is much simpler and easier to understand in only two dimensions.

4.1.1 Conversion of 2D Parametric CAD Geometry to Tessellated Geometry

In order to create a tessellated offset, the 2D profile must first be converted to tessellated geometry. The tessellation of the input curves in done in NX 6.0. The curves (lines, arcs, splines, etc.) of the profile are converted to a series of line segments as shown in figure 4.1.

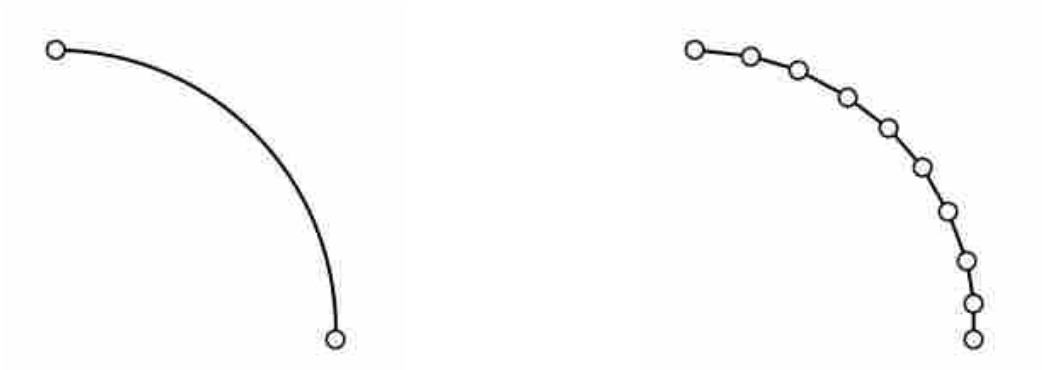


Figure 4-1: Curve Tessellation

Line segments are defined by two nodes and each node is defined by 2D coordinates (x, y). Both the line segments and the nodes also store information about which geometric entity they are associated with. The geometric information is necessary for later steps of the process.

```
Class Node2D
{
  Double x;
  Double y;
  Geometry2D* geometry;
};

Class Segment2D
{
  Node2D* node0;
  Node2D* node1;
  Geometry2D* geometry;
};
```

4.1.2 Signed Distance Field Creation

Once the geometry is converted to tessellated form and read into the software, the signed distance field is created. The distance field is created by computing the bounding box of the shape, increasing the size of the bounding box to accommodate the offset, and then filling the bounding box with a regular grid of points.

```
CreateGrid
{
   Box = Shape->GetBoundingBox();
   Box->IncreaseBoxSize(OffsetDistance*1.25);
   N = Box->LengthInX / GridSize;
   M = Box->LengthInY / GridSize;
   For(i=0; i<N; i++)
   {
      For(j=0; j<M; j++)
      {
         X = Box->MinX + i* GridSize;
         Y = Box->MinY + j* GridSize;
         GridPoint = New Point(X, Y);
         SetGridPointData(GridPoint);
      }
   }
}
```

At each grid point the minimum distance is calculated as well as whether the point is inside or outside the shape. If the point is inside the shape the distance is set to be negative. A value, alpha, is set for each grid node by subtracting the desired offset distance from the minimum distance. Alpha will therefore be positive for points that are outside the offset and negative for points that lie inside the offset.

```
SetGridPointData(Point)
{
   Distance = Shape->GetMinDist(Point);
   If(Shape->IsPointInSide(Point))
      Distance = - Distance;
   Point->Alpha = Distance - OffsetDistance;
}
```

Grid points are combined into groups of three forming triangle shaped cells. Figure 4.2 shows the distance field process. The grid points are created, the distance and sign is computer for each grid point, and cells are created from the grid points.

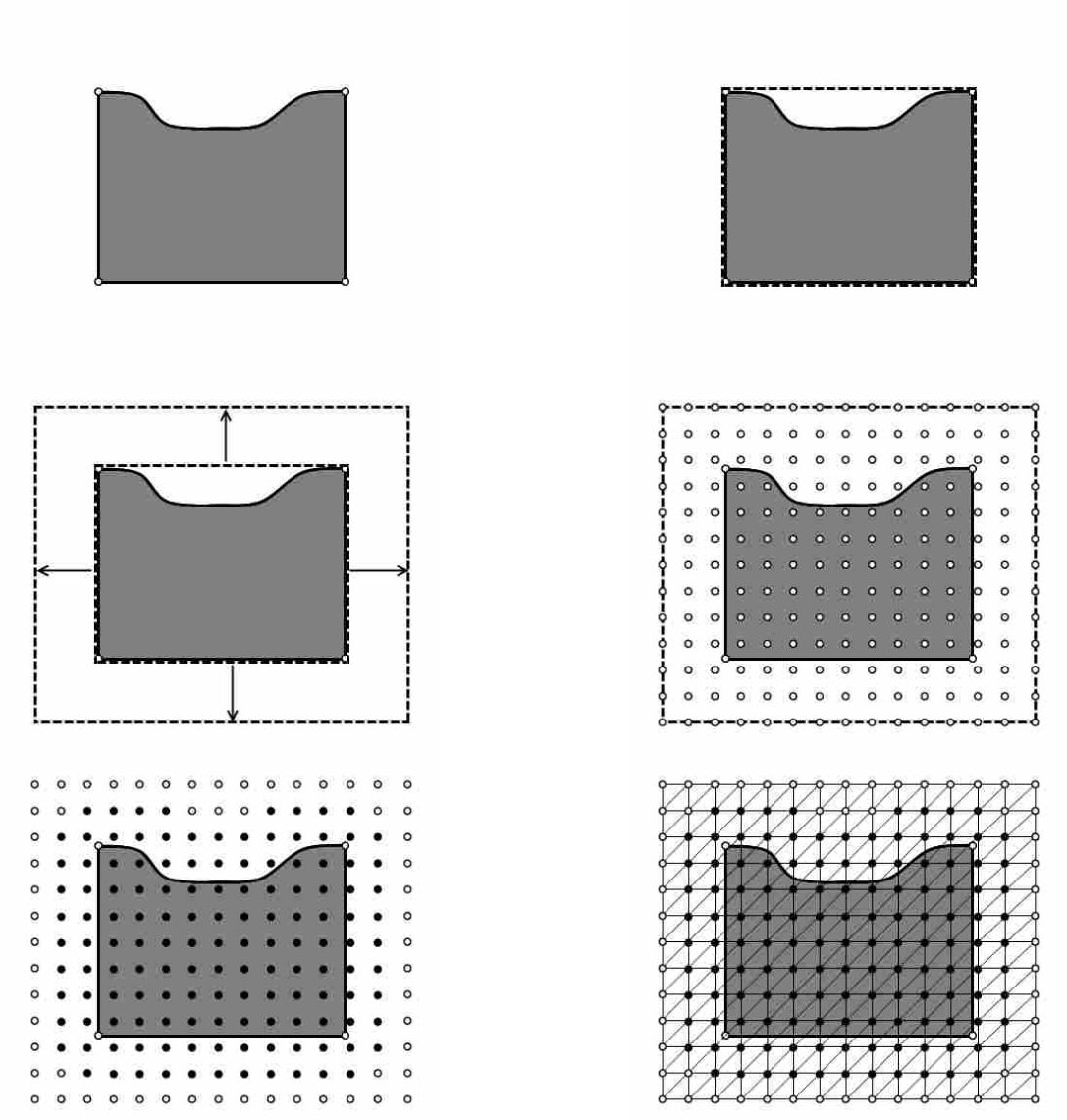


Figure 4-2: Signed Distance Field

4.1.3 Marching Triangles

Once the distance field is created, the tessellated offset can be created by contouring the distance field. The software relies on the algorithm Marching Triangles, which is a simplified

version of Marching Cubes. Because a triangle only has three points and each point can be in or out, there are only eight possible configurations which by symmetry can be paired down to just two. Figure 4.3 shows the two possible configurations for marching triangles.

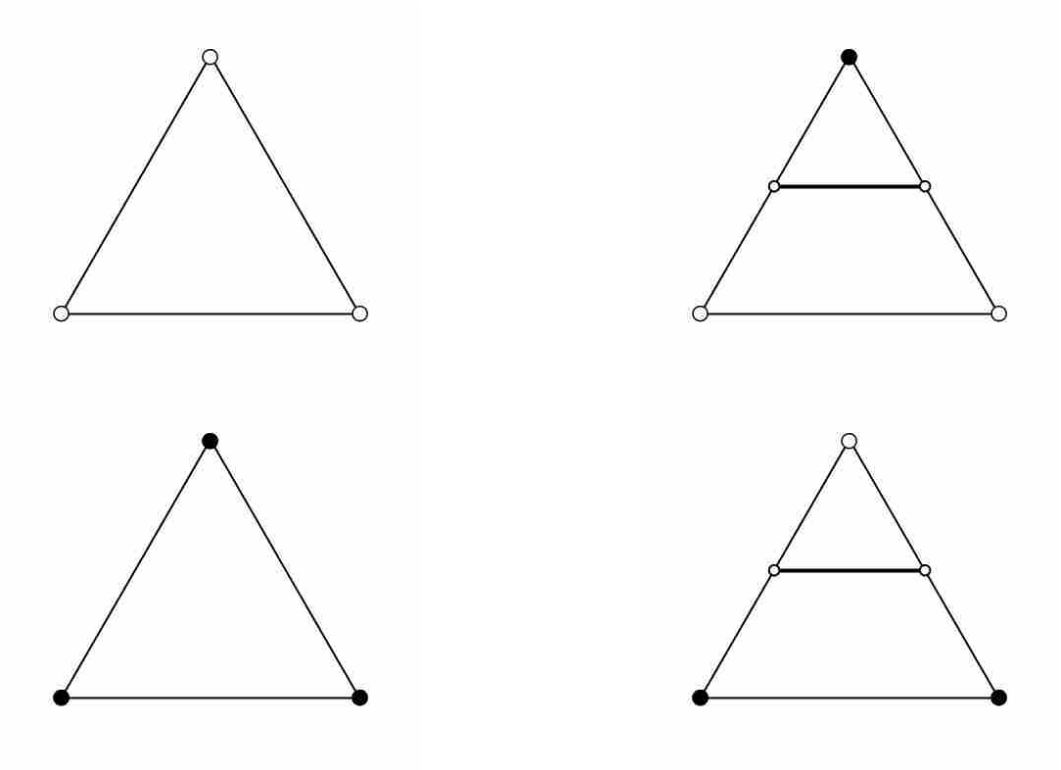


Figure 4-3: Marching Triangles

Each cell of the distance field is processed independently of the others to create the tessellated offset. As can be seen in figure 4.4, cells that have the same sign for all three nodes (i.e. all inside or all outside) do not contain the offset. Cells, however, with different signs at the nodes are used to create line segments that define the tessellated offset.

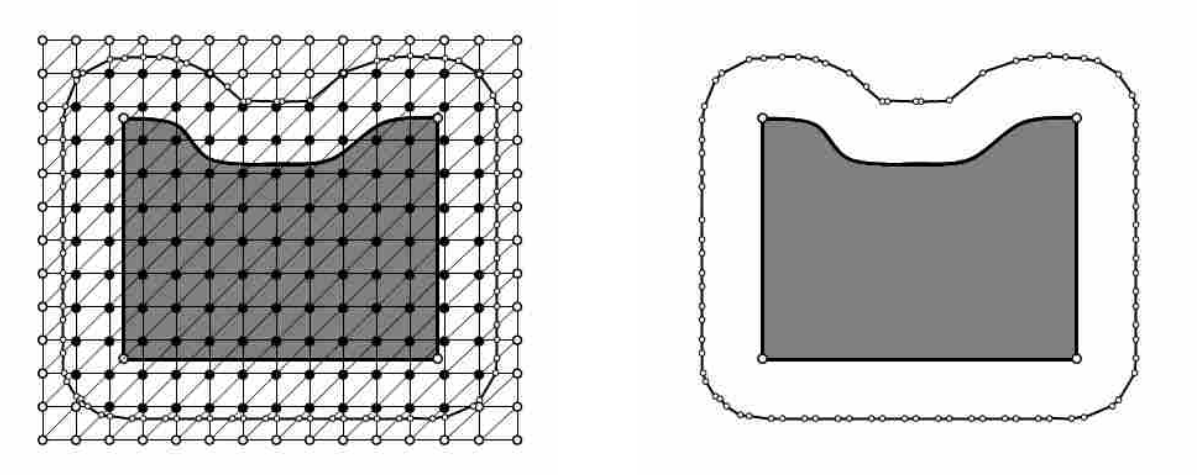


Figure 4-4: Contouring

4.1.4 Geometry Projection

At this point in the software, the tessellated offset has been created and is now ready to be converted back to parametric CAD geometry. The first step of converting back to parametric curves is to project each node of the tessellated offset back to the original shape as in figure 4.5. The geometry closest to the node is stored in the node.

```
ProjectNodes()
{
   For(i=0; i<AllOffsetNodes; i++)
   {
      ClosestElement = Shape->GetClosestElement(Node(i));
      Node(i)-> geometry = ClosestElement-> geometry;
   }
}
```

4.1.5 Line Segment Splitting

After the geometry of each offset node has been set, each line segment of the tessellated offset can be analyzed. The two nodes of a line segment are compared to each other. If both nodes have the same geometry the line segment is assumed to have that geometry as well. However, if the nodes of a line segment have different geometries then the line segment is split

and a new node is created. This new node is marked as a vertex and the two new line segments are marked with the geometry of the original nodes they connected to as shown in figure 4.5.

```
SplitLineSegment()
{
   For(i=0; i<AllOffsetSegments; i++)
   {
      Node0 = Segment(i)->Node0;
      Node1 = Segment(i)->Node1;
      If(Node0->Geometry == Node1->Geometry)
           Segment(i)->Geometry = Node0->Geometry;
      Else
         Vertex = (Node0 + Node1) / 2;
         NewSegment0 = CreateSegment(Node0, Vertex, Node0->Geometry);
         NewSegment1 = CreateSegment(Node1, Vertex, Node1->Geometry);
   }
}
```

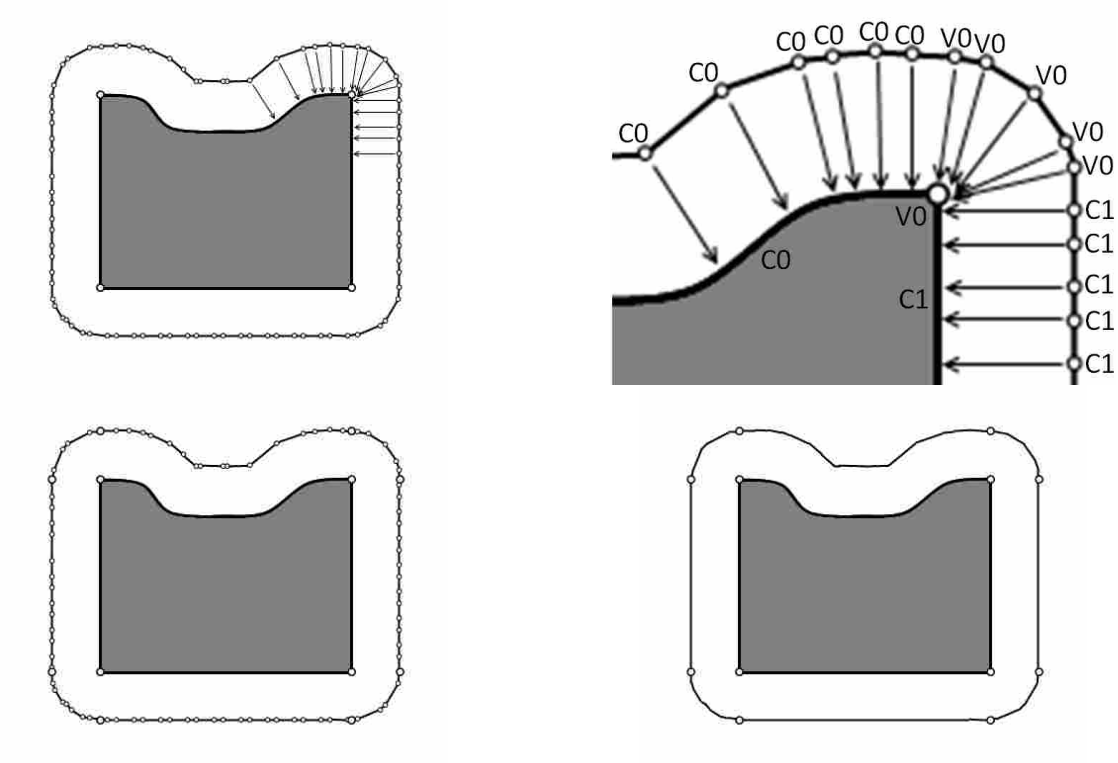


Figure 4-5: Geometry Projection

4.1.6 Recreation of Curves

At this point in the software, all nodes and line segments of the tessellated offset have been analyzed and assigned a geometric entity. The line segments can now be grouped together into regions. If two line segments share a node and that node is not marked as a vertex then the two line segments are part of the same region corresponding to a geometric entity.

```
GroupSegmentsTogether(Segment)
{
   SegmentGroup->Geometry = Segment->Geometry;

   While(true)
   {
      If(Node0->IsVertex)
         Break;
      Else
         SegmentGroup->AddFront(Segment);
         Segment = Segment->Previous;
   }

   While(true)
   {
      If(Node1->IsVertex)
         Break;
      Else
         SegmentGroup->AddBack(Segment);
         Segment = Segment->Next;
   }
}
```

For regions of line segments that came from simple geometric entities such as lines or arcs, the offset is easily created. The region of line segments is simply replaced by the true offset of the entity and the bounding vertices of the region are used to bind the parametric curve. The offset for a line is obtained by simply translating the line in the offset direction, and the offset of an arc/circle is obtained by simply increasing or decreasing its radius. For a free-form curve, however, the true offset cannot be used and must be approximated.

```
ReplaceLineSegmentRegionWithGeometry()
{
   For(i=0; i<AllLineSegmentRegions; i++)
   {
      Type = Regions(i)->GetGeometry->GetType;
      If(Type == SPLINE)
         ApproximateSpline(Regions(i));
      Else
         Offset(Regions(i));
   }
}
```

The region of line segments corresponding to the free-from curve can be used to create a NURBS curve that approximates the offset. The line segments are first converted into a degree one NURBS curve with knot vector spacing that corresponds to the lengths of the line segments. This allows for equally spaced points to be created from equally spaced parameters. These points are then used as the control points of a degree 3 curve that approximates the offset curve as shown in figure 4.6. The software uses the equally spaced points as control points but it is possible to create a NURBS curve that interpolates all of the equally spaced points. Using the points as control points was implemented because it results in a similar curve if there are sufficient points and also has the advantage of reducing noise in the curve.

```
ApproximateSpline()
{
   Spline->Degree = 1;
   Spline->NumberOfControlPoints = RegionNodes.Size;
   Length = 0.0;
   For(i=0; i<RegionNodes.Size; i++)
   {
      Length = Length + RegionNodes(i)->Distance(RegionNodes(i-1));
      Spline->Knots(i) = Length;
      Spline->Poles(i) = RegionNodes(i);
   }

   ApproxSpline->Degree = 3;
   ApproxSpline->NumberOfControlPoints = N;
   ApproxSpline->Poles = Spline->GetEvenlySpacedPoints(N);
   ApproxSpline->Knots->Uniform;
}
```

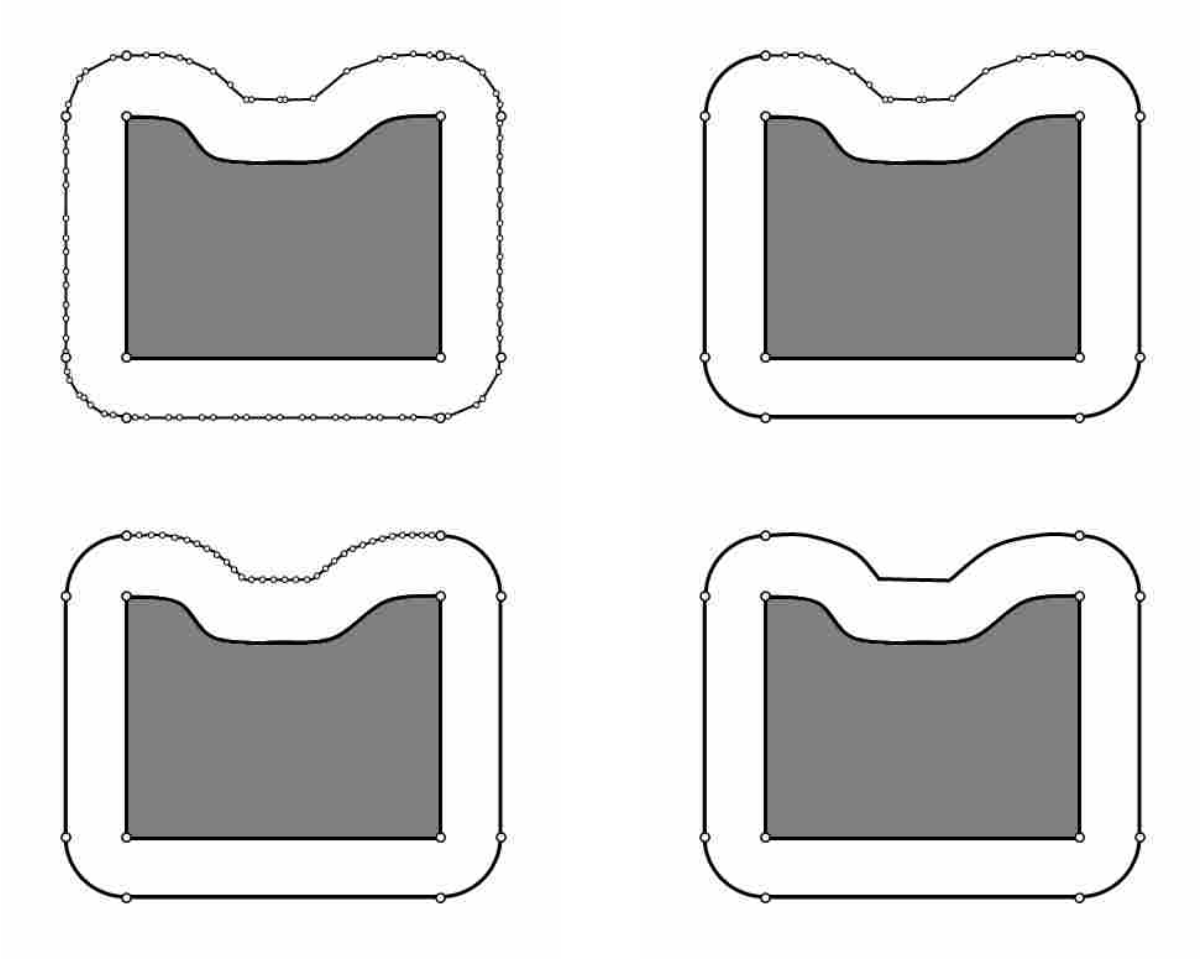


Figure 4-6: Tessellation to CAD

4.1.7 Final Product

Once all regions of line segments have been replaced by parametric CAD geometry the offset is complete. The end result is a self-intersection free offset comprised of parametric curves. No self-intersection tests were performed and no trimming was required either. The results of the software can be seen in chapter 5.

4.2 Implementation in 3D

The 3D version of the offset method is significantly more challenging to implement. Some aspects of the software are easily transferred from 2D to 3D however in general the 3D method is more complex. Instead of creating planar offset curves, 3D offset surfaces must now be created.

4.2.1 Conversion of 3D Parametric CAD Geometry to Tessellated Geometry

In order to generate the tessellated offset surface in 3D, it is first necessary to convert the 3D parametric CAD geometry to tessellated geometry. For the purpose of this research the built in functionality of the CAD system NX 6.0 is used to create this tessellation. As previously mentioned, tessellating of CAD geometry is a well understood process that all CAD systems rely on to display geometric data to the computer's monitor.

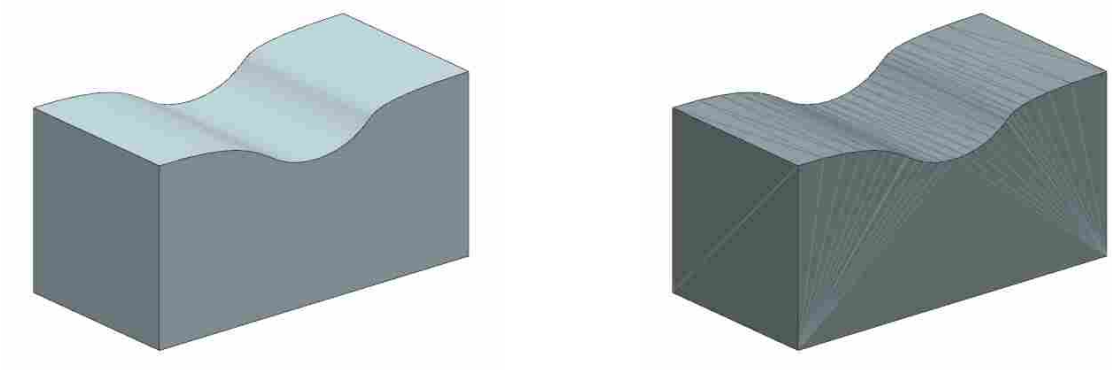


Figure 4-7: Surface Tessellation

4.2.2 Tessellation Storage in Half-Edge Data Structure

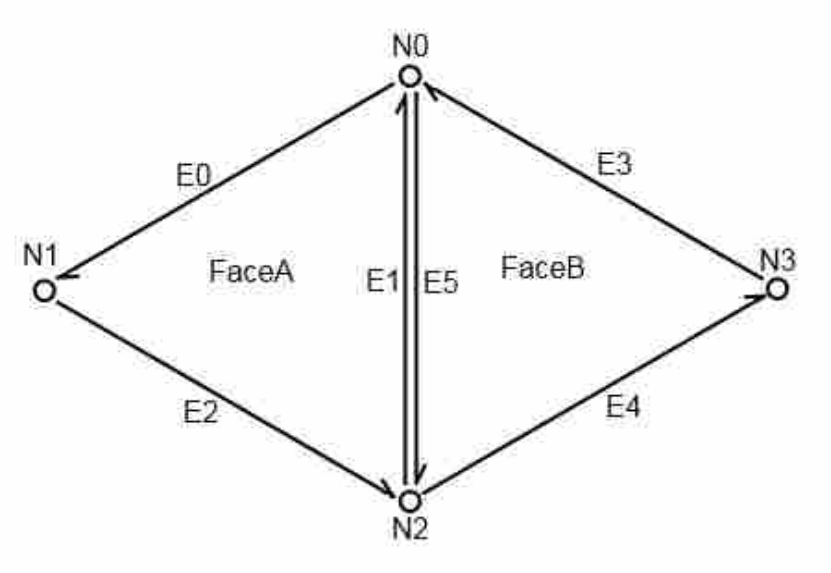
Both the original tessellated geometry and the tessellated offset surface must be stored in an efficient manner that allows for easy querying of the data. There are many methods for storing

tessellated data but the method that has proven the most useful for this project is the Half-Edge Data Structure (HEDS). The HEDS model breaks the tessellation down into three primitives. First, the node, which is a point location in space. Second, the half-edge which is an edge that connects two nodes. And third, a face that is the collection of three half edges. Half-edges are able to have a direction associated with them. As seen in figure 4.8 in order for faces A and B to have consistent winding the shared edge must point in two different directions. By splitting the edge in two, each edge can have the correct direction for its corresponding face.

```
Node
{
   X,Y,Z;
   U,V;
   Edges;
   Geometry;
}

HalfEdge
{
   Node;
   NextEdge;
   PrevEdge;
   OppositeEdge;
   Face;
   Geometry;
}

Face
{
   Edge;
   Geometry;
}
```



Edge 1 Data:
Node = N2
Opposite = E5
Next = E0
Prev = E2
Face = FaceA

Edge 5 Data:
Node = N0
Opposite = E1
Next = E4
Prev = E3
Face = FaceB

Figure 4-8: Half-Edge Data Structure

4.2.3 Signed Distance Field Creation

A volumetric approach, similar to the 2D method, is used to create the tessellated offset surface. The volumetric approach can easily handle topological changes and self-intersections making it the most robust method. This method relies on the creation of a signed distance field. The distance field is a set of points equally spaced around the part. The distance and sign of each one of these points is then calculated. Unlike the 2D method that only requires a planar grid, the 3D method must add another dimension.

```
CreateGrid()
{
   Box = Shape->GetBoundingBox();
   Box->IncreaseBoxSize(OffsetDistance*1.25);
   N = Box->LengthInX / GridSize;
   M = Box->LengthInY / GridSize;
   P = Box->LengthInZ / GridSize;
   For(i=0; i<N; i++)
   {
      For(j=0; j<M; j++)
      {
```

```
        For(k=0; k<P; k++)
        {
           X = Box->MinX + i*GridSize;
           Y = Box->MinY + j*GridSize;
           Z = Box->MinZ + k*GridSize;
           GridPoint = New Point(X, Y, Z);
           SetGridPointData(GridPoint);
        }
     }
  }
```

4.2.4 Marching Tetrahedron

Once the distance and sign of all points in the grid have been calculated, the signed distance field can be contoured. In 2D the distance field consists of triangle shaped cells and relies on Marching Triangles to create line segments. Now the Marching Tetrahedron algorithm is used to extract a tessellated offset surface from the distance field.

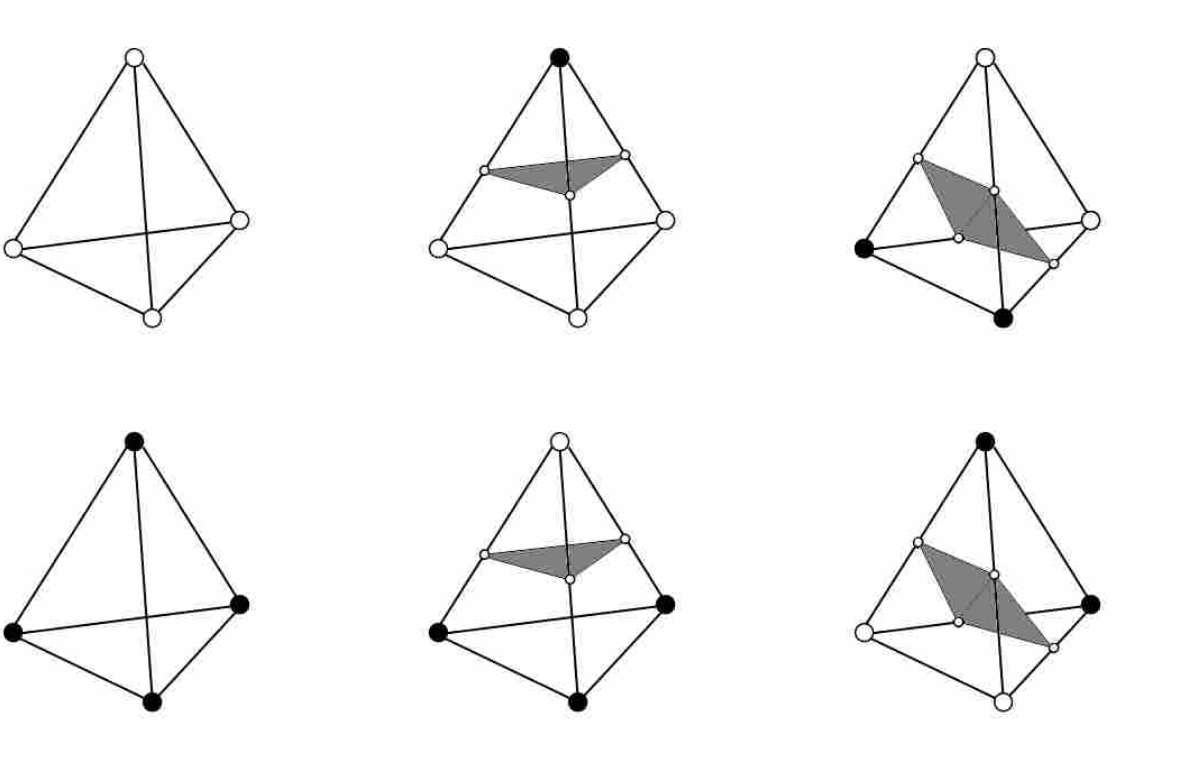


Figure 4-9: Marching Tetrahedron

This is done by grouping grid points into groups of four (forming tetrahedron) and comparing the signs at each point. Because there are four points and each point can be either in or out, there are 16 possible configurations. These configurations can be combined through symmetry down to just three cases. These cases are shown in figure 4.9.

4.2.5 Geometry Projection

Now that the tessellated offset surface is created, the geometry of the original shape can be projected onto the offset. Each node of the offset surface is projected back to the original shape, or in other words, the minimum distance is found between the node and the original shape. The nearest primitive is found and its geometry is stored in the node of the offset surface.

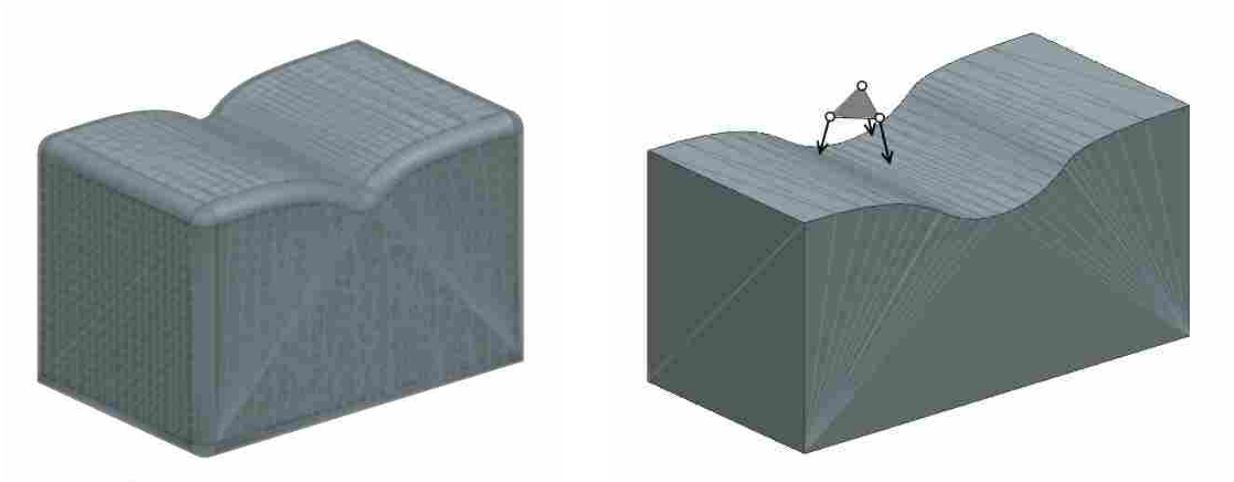


Figure 4-10: Geometry Projection

Once the closest geometry of all nodes has been set, each face of the offset surface is analyzed. For faces where all three nodes have the same geometry, the face is also assumed to have that same geometry. However, faces where the three nodes do not share the same geometry, additional analysis is required.

```
AnalyzeFaces()
{
   For(i=0; i<AllFaces; i++)
   {
      Node0 = AllFaces(i)->Edge->Node;
      Node1 = AllFaces(i)->Edge->NextEdge->Node;
      Node2 = AllFaces(i)->Edge->NextEdge->NextEdge->Node;
      If(Node0->Geometry == n1->Geometry && Node1->Geometry == Node2->Geometry)
         AllFaces(i)->Geometry = Node0->Geometry;
      Else
         SplitEdge(Node0, Node1);
         SplitEdge(Node1, Node2);
         SplitEdge(Node2, Node0);
         SplitFace(AllFaces(i));
   }
}
```

Each edge of the face is checked to see if there is a change in geometry, if so, the bisection method is used to determine where the edge should be split. It is possible for the edge to have more than one location where its geometry changes. These changes are associated with that edge and stored. Once all three edges of the face have been analyzed the face can be subdivided.

```
SplitEdge(NodeA, NodeB)
{
   While(true)
   {
      If(NodeA->Distance(NodeB)  < EdgeTolerance)
         Break;
      TestNode = (NodeA + NodeB)/2.0;
      TestNode->Geometry = Shape->GetClosestElement->Geometry;
      If(TestNode->Geometry = NodeA->Geometry)
         NodeA = TestNode;
      Else If(TestNode->Geometry = NodeB ->Geometry)
         NodeB = TestNode;
      Else
         SplitEdge(NodeA, TestNode);
         SplitEdge(TestNode, NodeB);
   }
   Crossing->Node = (NodeA + NodeB)/2.0;
   Crossing->GeometryA = NodeA->Geometry;
   Crossing->GeometryB = NodeB->Geometry;
}
```

4.2.6 Face Subdivision by Modified Ear Clipping

The method for splitting the face into sub triangles is a modified version of the ear clipping algorithm. The ear clipping algorithm is the process of taking a polygon and triangulating the interior by forming triangles from three consecutive points on the polygon. In the regular ear clipping method care must be taken that the formed triangles do not contain any other point on the polygon. Because the face being divided is completely convex this cannot happen. Instead, the criteria for creating sub triangles is that three consecutive nodes must have the same geometry. Each edge crossing corresponds to a change in geometry. Each crossing, therefore, has two geometries associated with it. The algorithm now searches the nodes in order and when three consecutive nodes are found that have the same geometry a triangle is formed and the middle node is removed from the list. This process is illustrated in figure 4.11.

```
SplitFace()
{
   Node0 = AllFaces(i)->Edge->Nod
   Node1 = AllFaces(i)->Edge->NextEdge->Node;
   Node2 = AllFaces(i)->Edge->NextEdge->NextEdge->Node;

   Crossings.Add(GetCrossings(Node0, Node1));
   Crossings.Add(GetCrossings(Node1, Node2));
   Crossings.Add(GetCrossings(Node2, Node0));

   While(TRUE)
   {
      For(i=0; i<Crossings.Size; i++)
      {
         NodeA = Crossings(i);
         NodeB = Crossings(i+1);
         NodeC = Crossings(i+2);
         If(FindCommonGeometry(NodeA, NodeB, NodeC, Geom))
         {
            NewFace = CreateFace(NodeA, NodeB, NodeC);
            NewFace->Geometry = Geom;
            Crossings.Erase(i+1);
            Break;
         }
```

```
        }

        If(Crossings.size < 3)
            Break;
        If(NewFace == NULL)
            CreateVertex(Crossings);
            Break;
    }
}
```

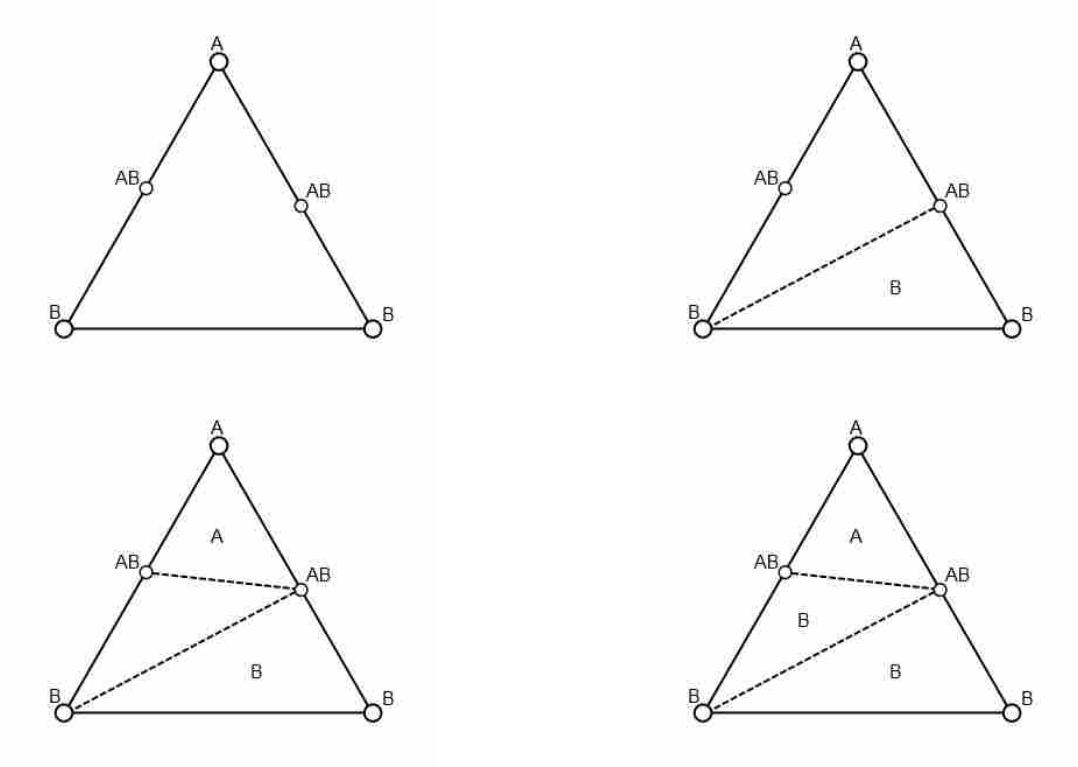


Figure 4-11: Modified Ear Clipping

The ear clipping process continues until either the list is empty, or until no more triangles can be created. If no more triangles can be created and the list is not empty a vertex must be created. This is done by inserting a node into the remaining region. A node is created at the average location of all the remaining nodes and a triangle fan is created using the new node and the list of nodes. The geometry of each face is set by comparing two consecutive nodes and finding which geometry they have in common.

```
InsertVertex(Crossings)
{
   Vertex(0.0, 0.0, 0.0);
   For(i=0; i<Crossings.Size; i++)
      Vertex = Vertex + Crossings(i);
   Vertex = Vertex / Crossings.Size;
   Vertex->Geometry = Vertex->Point;

   For(i=0; i<Crossings.Size; i++)
   {
      NodeA = Crossings(i);
      NodeB = Crossings(i+1);
      Face = NewFace(NodeA, NodeB, Vertex);
      Face->Geometry = CommonGeometry(NodeA, NodeB);
   }
}
```

Once all faces have been analyzed and split, the tessellated offset surface is ready to be converted back to parametric CAD geometry. Parametric CAD geometry is stored in the BREP model as detailed in chapter 2. There are three types of geometry required to construct the BREP model, points, curves and surfaces. Therefore, in order to fully define the geometry, all three geometry types must be created.

4.2.7 Recreation of Vertices

The first type of geometry to be recreated is a vertex. What identifies a vertex of the surface is a node that has more than two geometries associated with its neighboring faces. However, this only occurs at the new nodes that were inserted by the Ear Clipping algorithm. By storing the new nodes that were created, all vertices have already been found. These nodes are marked as vertices of the BREP model and the geometry of the vertex is simply the point location of the node.

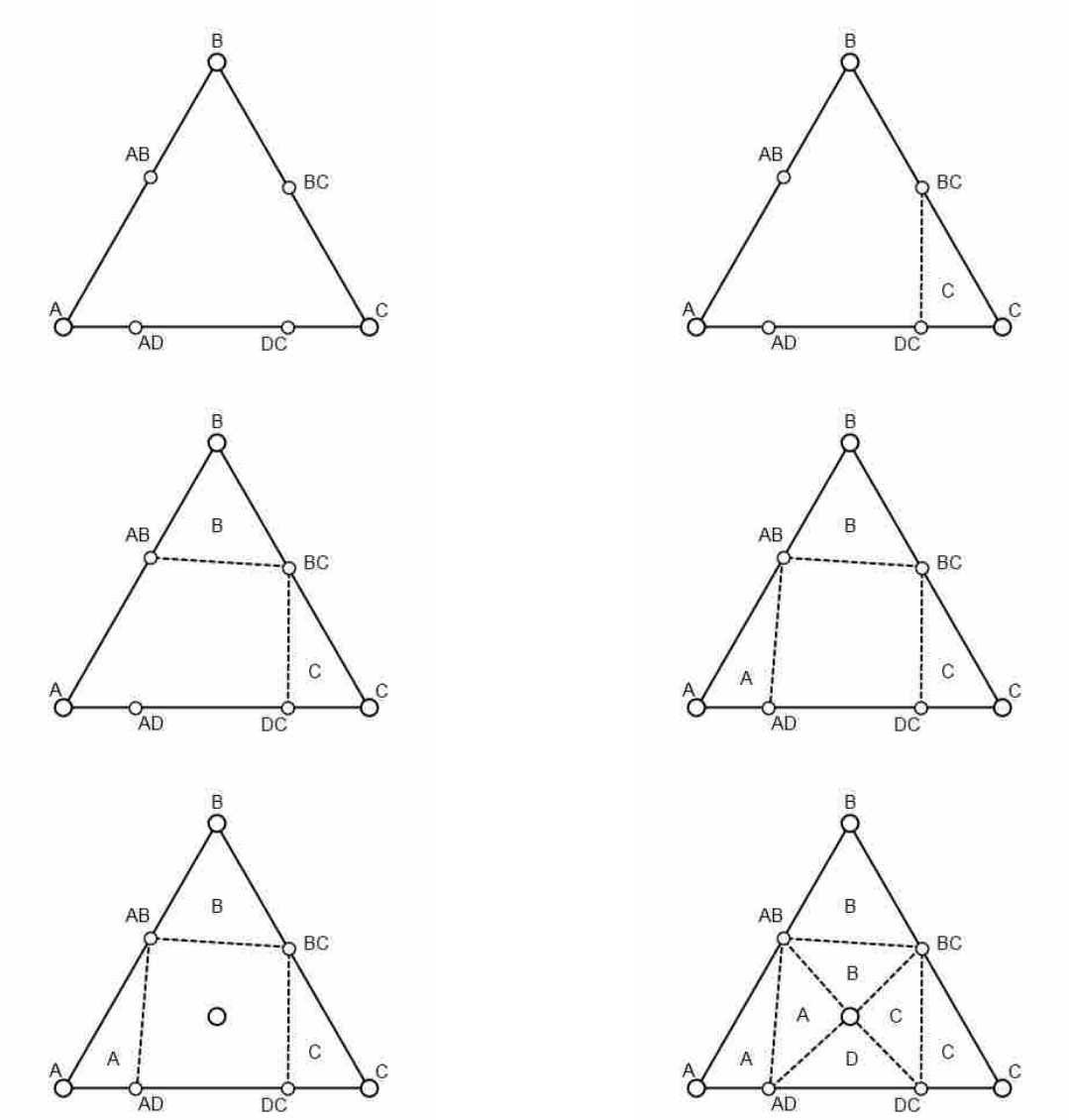


Figure 4-12: Vertex Creation

4.2.8 Recreation of Edges

All half-edges of the offset surface are examined, if the opposing faces have the same geometry the half-edges are marked with that geometry. If the opposing faces have different geometry then the half-edges are marked as a geometric edge. These half-edges are combined end to end to form half-edge groups. These groups are used to create parametric curves. Using the same algorithm detailed in the 2D section, the edges are converted to a degree one NURBS

curve that has a knot vector corresponding to the half-edge lengths. Equally spaced points are then created and used as control points to create a degree three NURBS curve. As mentioned in previously, using these points as control points has an advantage over interpolating the points.

4.2.9 Recreation of Faces

By far the most challenging portion of the process is the recreation of surfaces. Similar to how simple curves in the 2D case can be offset directly, simple curves and surfaces in the 3D case can also be offset directly. Below is a table of showing how simple shapes are offset directly.

For free-form surfaces, the underlying offset surface cannot be calculated directly and must be approximated. Unlike simple geometric surfaces, where the corresponding tessellated offset is not needed, the tessellated offset is of great value to the free-form surface reconstruction. The tessellated offset provides a guide of where the free-form surface should exist.

Table 4-1: Surface Offset Methods

Element	Offset
Vertex	Sphere with vertex as center and offset distance as radius.
Line	Cylinder with line as axis and offset distance as radius.
Arc/Circle	Torus with Arc/Circle as major radius and offset distance as minor radius.
Plane	Plane moved in direction of normal the offset distance.
Cylinder	Cylinder with radius +/- offset distance depending on if offset is inside or outside the cylinder.
Sphere	Sphere with radius +/- offset distance depending on if offset is inside or outside the sphere.
Cone	Cone with apex translated +/- depending on if offset is inside or outside of cone.
Torus	Torus with minor radius +/- offset distance depending on if offset is inside or outside the torus.
NURBS Surface	Approximated NURBS surface.
NURBS Spline	Approximated NURBS surface.

In order to create a NURBS surface a rectangular grid of control points is needed. The simplest way to apply a rectangular grid of points to the tessellated offset surface is to first convert the tessellated surface to two dimensions. Several methods were attempted but the most robust method is by using what is called a Least Squares Conformal Map (Levy 2002). This process takes a tessellated surface in 3D and flattens it to 2D with a minimum amount of distortion. Implementing a custom version of the Least Squares Conformal Map is beyond the scope of this research, therefore, the open source library CGAL was used to implement this method.

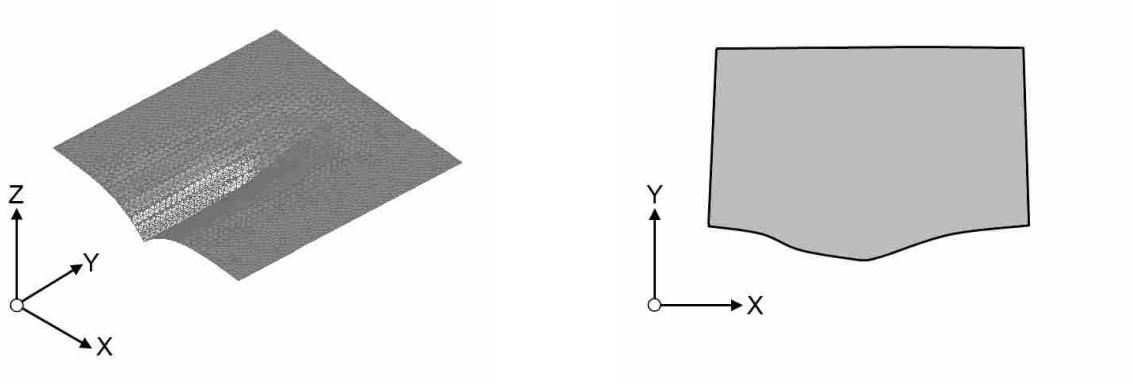


Figure 4-13: Parameterization of Tessellated Surface

It is important to note that not any tessellated surface can be converted to 2D. For a surface to be represented in 2D the surface must have a genus of 0. The genus of a surface may be thought of as the number of times a surface must be cut in order to lay it flat. Figure 4.14 shows two surfaces. Part A shows a surface of genus 0, this surface may be converted to 2D using the Least Squares Conformal Map method. Part B shows a surface that has a genus of 1 and cannot be converted to 2D. In order for the Least Squares Conformal Map method to work on this surface, it must first be cut and made into a surface of genus 0.

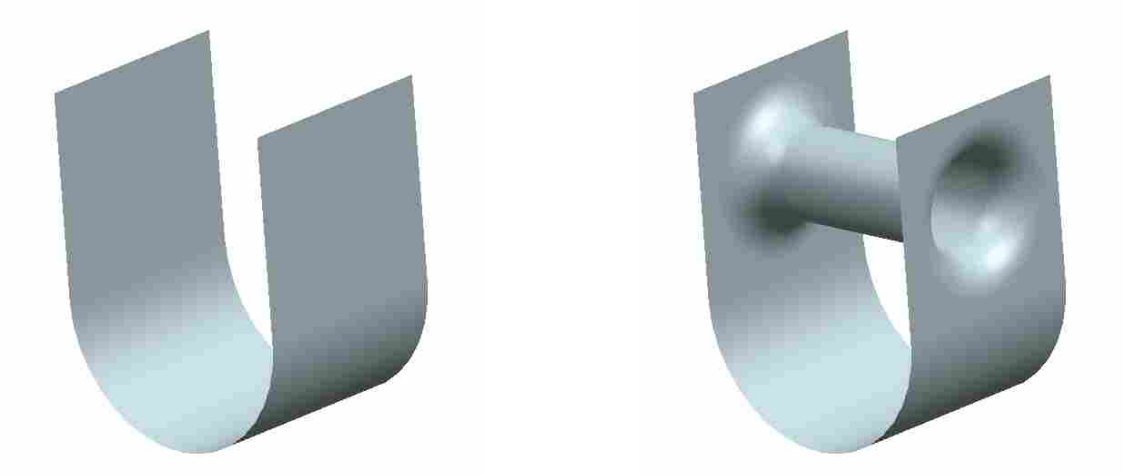


Figure 4-14: Surface Genus

In order to cut the surface, the underlying geometry is required. The points of the mesh are projected back to the underlying surface and the parameter values of each point are stored. The edges are then analyzed and the edge with the largest 2D length is marked as part of the global intersection. The edge is then used as a starting point and a marching algorithm is used to create the cut.

Once the tessellated surface is parameterized, each node contains both a 3D location (x,y,z) as well as a 2D location (u,v). With a 2D location defined for each node the tessellated surface can be drawn to a plane. From the 2D points a 2D bounding box is calculated. Grid spacing is chosen based on the average length of the half-edges. From this bounding box and grid size, a regular grid is created over the 2D domain. Each point is checked to see if the point lies inside or outside the surface boundaries. For points that lie inside the boundaries the 3D position may be calculated.

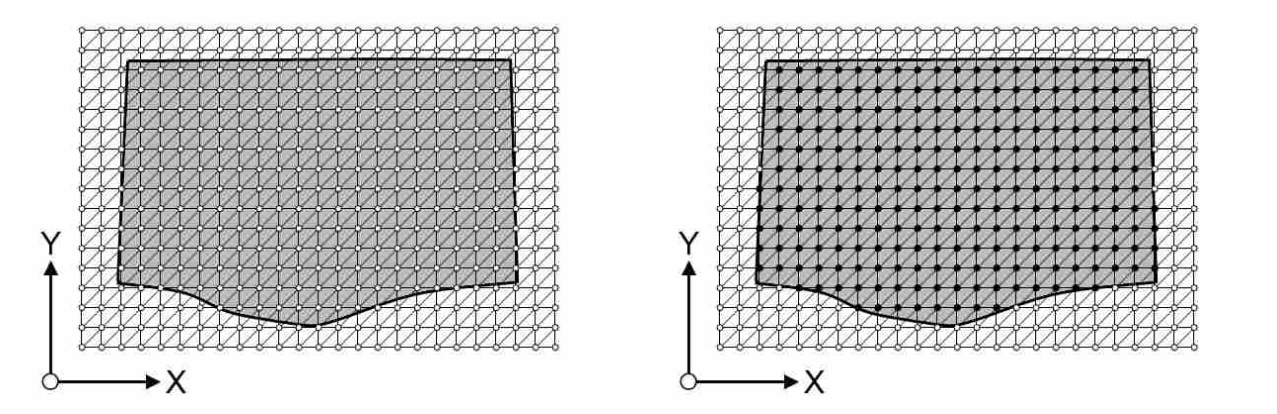


Figure 4-15: Control Point Grid

In order to create a NURBS surface all points of the grid must be positioned. For points that lie outside of the boundaries their positions must be extrapolated from their neighboring points. If a neighboring face is completely defined it can be used to calculate the location of the point. The point location is set by mapping its 2D relationship to its neighboring faces into 3D. This allows the grid of points to be extended in order for a complete NURBS surface to be defined.

```
ExtrapolatePointLocation(Node)
{
   AvgPoints;
   For(i=0; i<Node->Edges.Size; i++)
   {
      Face = Node->Edges(i)->NextEdge->OppositeEdge->Face;
      NodeA = Face->Edge->Node;
      NodeB = Face->Edge->NextEdge->Node;
      NodeC = Face->Edge->NextEdge->NextEdge->Node;

      If(NodeA->IsSet && NodeB->IsSet && NodeC->IsSet)
      {
         Vector01 = NodeB – NodeA;
         Vector02 = NodeC – NodeA;
         Normal = Vector01.Cross(Vector02);
         Direction = Vector01.Cross(Normal);
         Param = Vector01.Dot(Node - NodeA)/ Vector01.Dot(Vector01);
```

```
            BasePoint = NodeA + (NodeB – NodeA)*Param;
            Length2d = NodeA.Distance2D(NodeB);
            Length3d = NodeA.Distance3D(NodeB);
            Distance2d = Node.Distance(BasePoint);
            Distance3d = Distance2d * Length3d / Length2d;
            NewPoint = BasePoint + Direction*Distance3d;
        }
    }
}
```

In many cases the extended portion of the surface becomes very poorly shaped. This however, is not a problem because the BREP model trims away the portion of the surface that lies outside the border. This algorithm generally results in a well-conditioned surface inside the boundaries and a poorly conditioned surface outside.

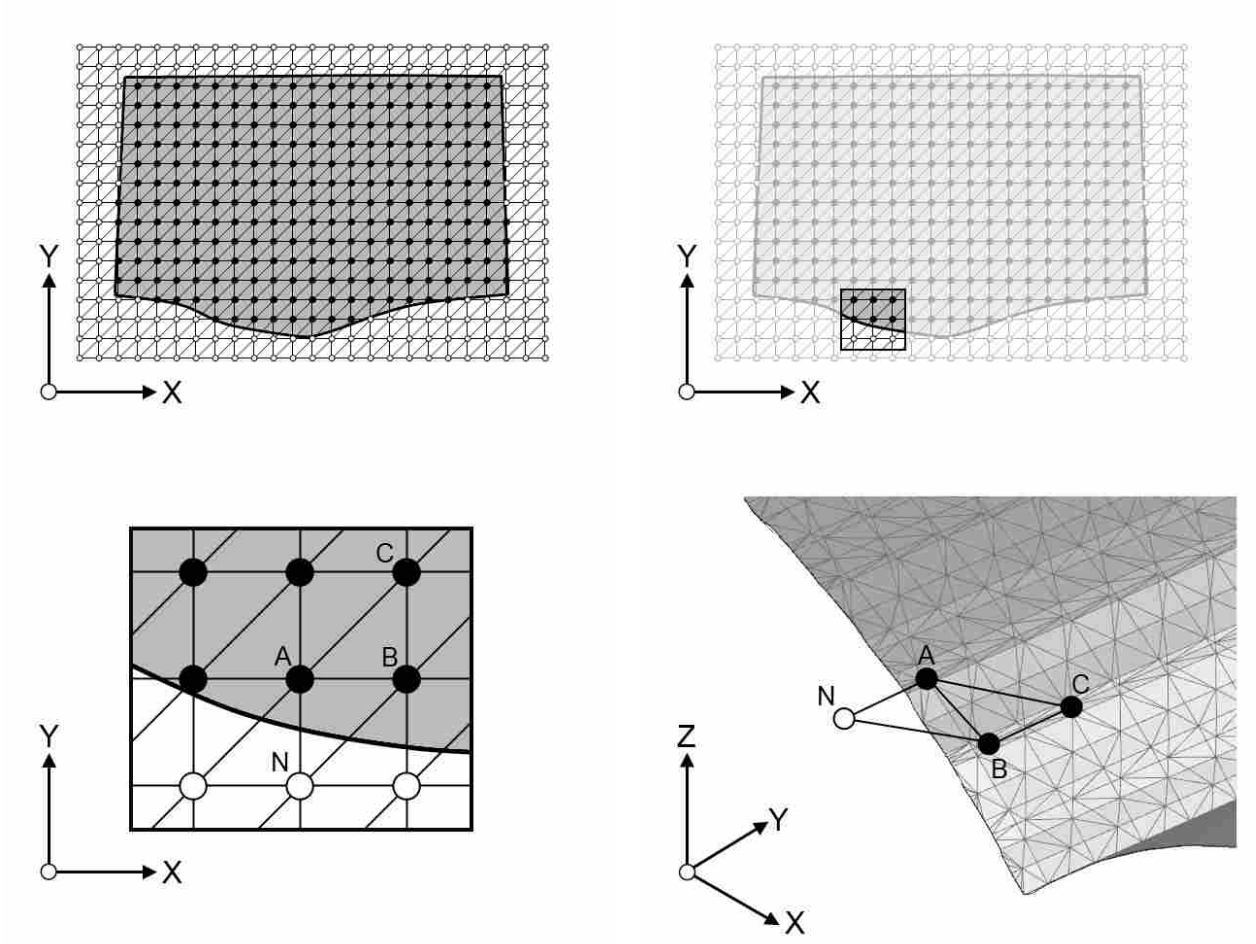


Figure 4-16: Extrapolating Point Locations

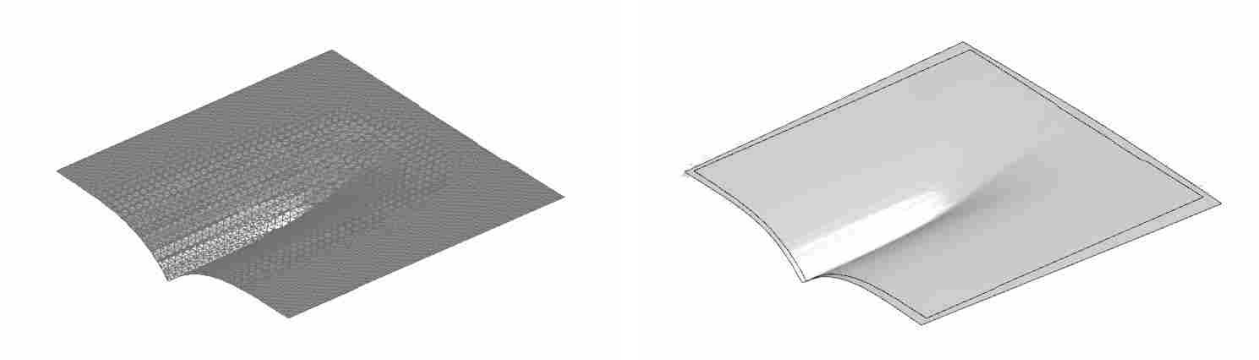


Figure 4-17: Tessellated Surface to NURBS

4.3 Putting it All Together

With all three geometric entities created, vertices, curves and surfaces, the BREP can be created. The vertices are used to define the end points of curves. The curves are used to trim the surfaces into faces. The faces can then be combined into a solid model. The results of this implementation will be discussed in the next chapter.

4.4 Accuracy of the Hybrid Method

Before discussing the results of this method, however, it is important to point out that this method is not without errors. This method introduces error at several key steps. The first place in the method where error is introduced is the tessellated body that is used to represent the part being offset. It is not necessary to use a tessellated version for distance queries but it is much faster. The error introduced by this approximation can be reduced by increasing the accuracy of the tessellation, or by performing distances queries directly on the original CAD geometry; both of these routes will increase the accuracy but will also increase the computation time.

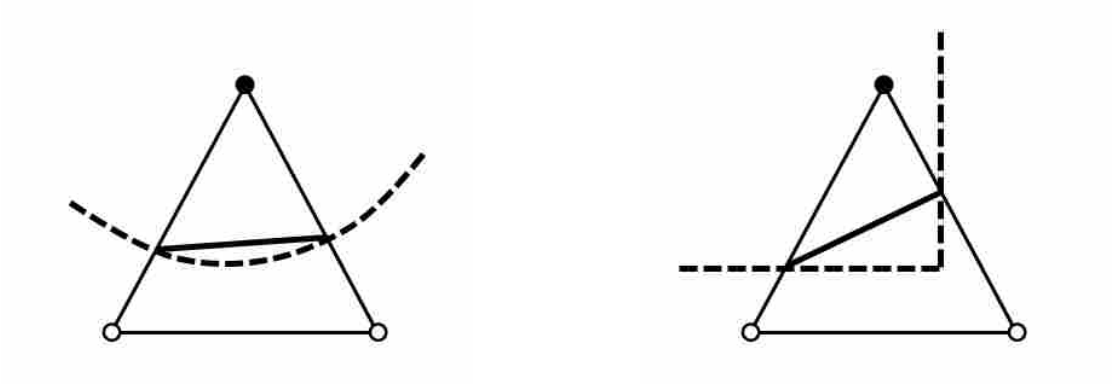


Figure 4-18: Error Introduced by Contouring

Once the distances are computed and stored in the nodes of the distance field, the offset surface is then approximated by contouring. Contouring will also introduce error into the approximation. Figure 4.18 demonstrates how error is introduced by contouring. As the tessellated offset is created in each cell of the distance field, the offset is approximated by straight lines (planar faces in 3D). In reality, the ideal offset surface (Figure 4.18 dashed line) may be curved (Figure 4.18 left) or may have a sharp corner (Figure 4.18 right) but is being approximated by a straight line. This is one of the draw backs to the Marching Tetrahedron method is that it does not do a good job of preserving sharp features. This error can be minimized by increasing the resolution of the distance field. An increase in resolution of the distance field, however, will also result in longer computation times.

With the surface contoured and subdivided, the individual surface patches can be recreated. Surface patches that are offsets of simple geometry (e.g. planes, cylinders, etc.) can be replaced with the actual offset surface which is trivial to calculate. The offset surface of the simple geometry therefore has no error. However, for the free-form surfaces, the surface patches are created by fitting a NURBS surface to the tessellated patch. More error is introduced by this fitting process, particularly near the borders of the patch where the surface will be trimmed. This

error can be minimized in the same way as before, by increasing the resolution. This time however, not only will increasing the resolution increase the computation time but will also increase the number of poles of the surface. This increase in poles leads to a much more complex surface, and much more data that must be stored by the part.

5 RESULTS

In order to demonstrate the value of the hybrid method, a series of tests were conducted. A set of test parts were created to be offset by the method. For comparison purposes, the same parts were offset by several commercial CAD systems. These test parts range from very simple parts that all CAD systems are able to offset to rather complex parts that none of the CAD systems are able to offset. The test parts are shown and the offsets generated by the various CAD systems are also presented. The results of the hybrid method are then shown and compared to the CAD system results.

5.1 Test Parts

The following figures show the CAD models that were used as the test cases for this method. Part 1 is a simple L shaped block that has been extruded. The second part is similar but has an added feature. Both test parts 1 and 2 do not contain any free-form surfaces. These test parts are the least complex and it was anticipated that all CAD systems and the hybrid method would be capable of successfully offsetting them.

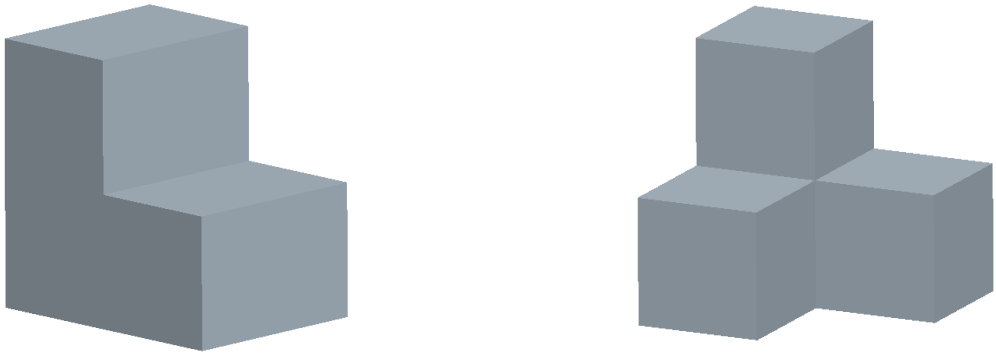


Figure 5-1: Simple Test Parts

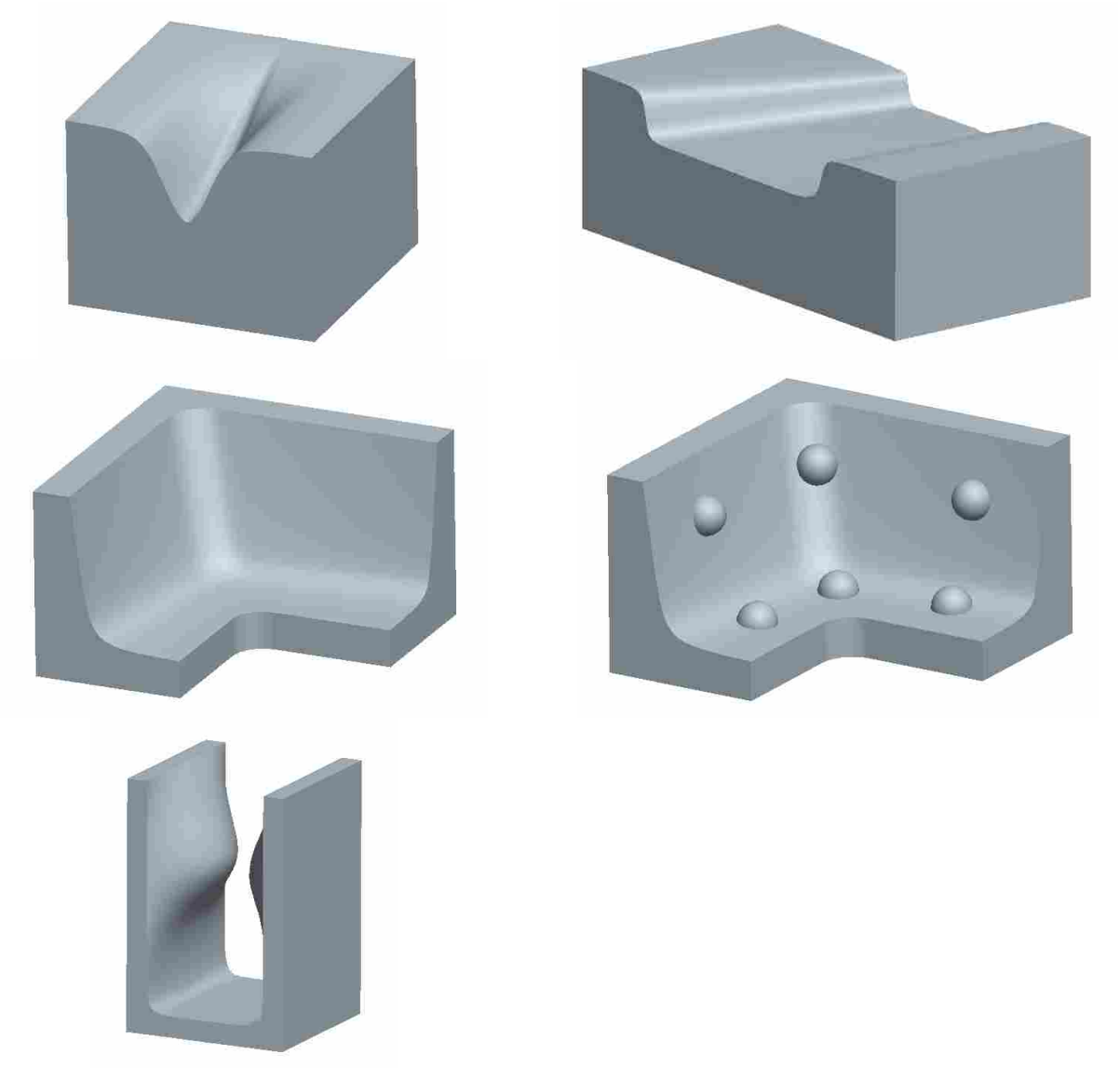


Figure 5-2: Complex Test Parts

The true measure of the offset function, however, is how the method deals with free-form surfaces. The complex test parts all have free-form surfaces and it was anticipated that they could not successfully be offset by the various CAD systems. Test parts 3, 4 and 5 all contain regions of high curvature with varying geometry. Test part 6 contains areas of high curvature but also will experience topological changes due to the added features. Test part 7 contains a free-form surface that will encounter a global self-intersection. These test parts provide a good sample of what problematic areas an offset tool will have to deal with in order to be robust.

5.2 Commercial CAD Systems

The CAD systems that were used in order to bench mark the hybrid method are: NX, Pro Engineer, CATIA, SolidWorks, and Inventor. These CAD systems represent the most recognized and widely used CAD systems in industry. All test parts are approximately 2 inches cubed in size. They are all tested at an offset distance of 0.5 inches, approximately 25%, which is a large offset distance.

5.2.1 NX Results

The following images show the results of offsetting the test parts by way of NX. As expected the two simple test cases were offset correctly. Test cases 3, 5, and 6 all failed to produce results for the specified offset distance. Test case 4 did produce a result but upon inspection, it can be seen that the offset surface intersects the original shape which makes for a very poor result. Test part 7 was offset correctly even with the global intersection.

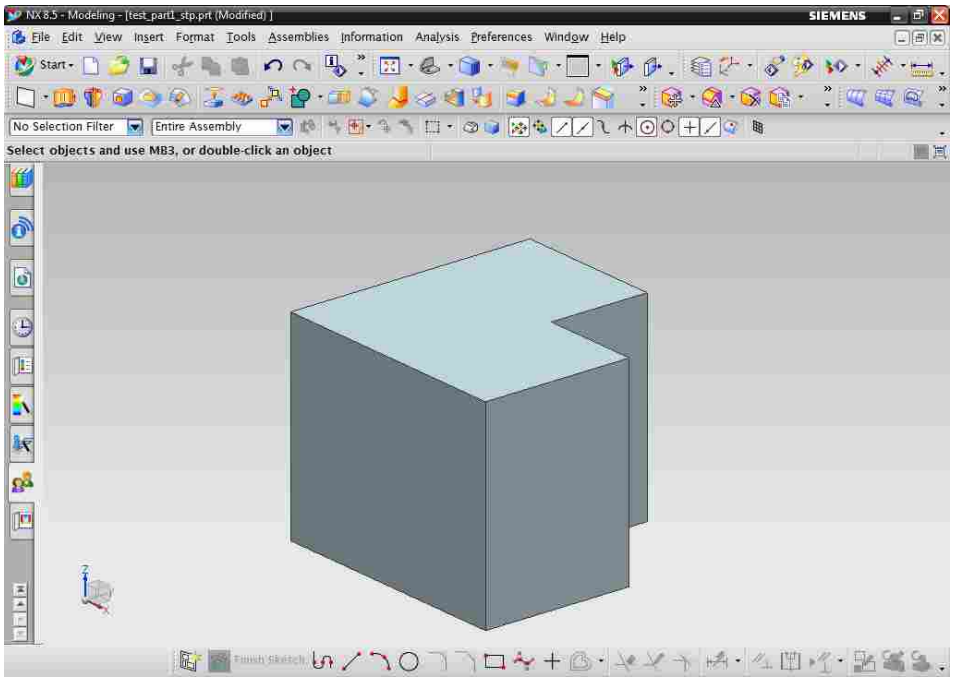


Figure 5-3: NX Part 1

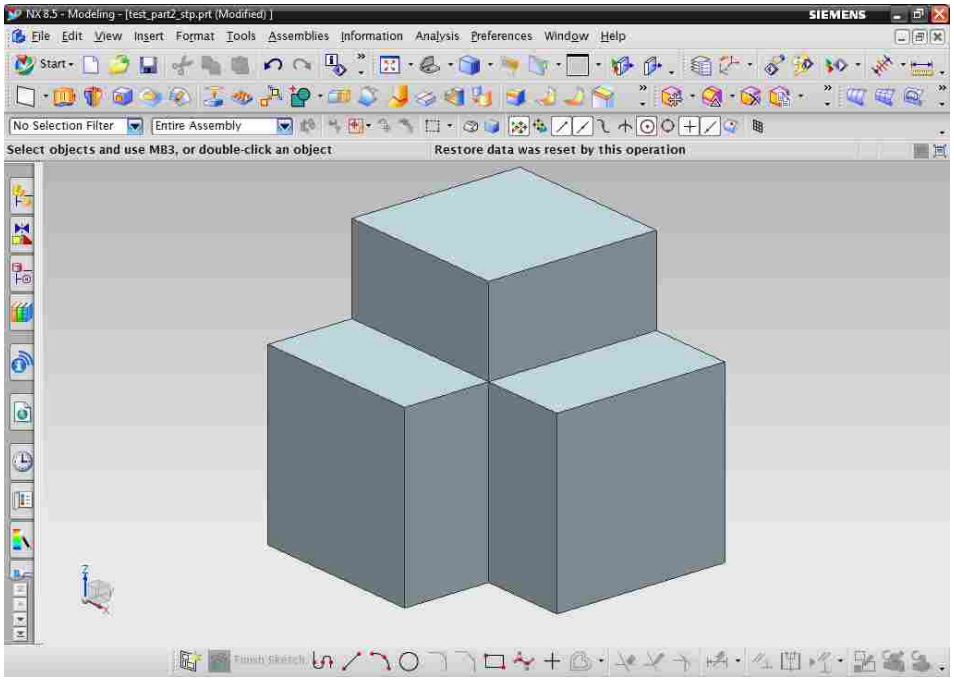


Figure 5-4: NX Part 2

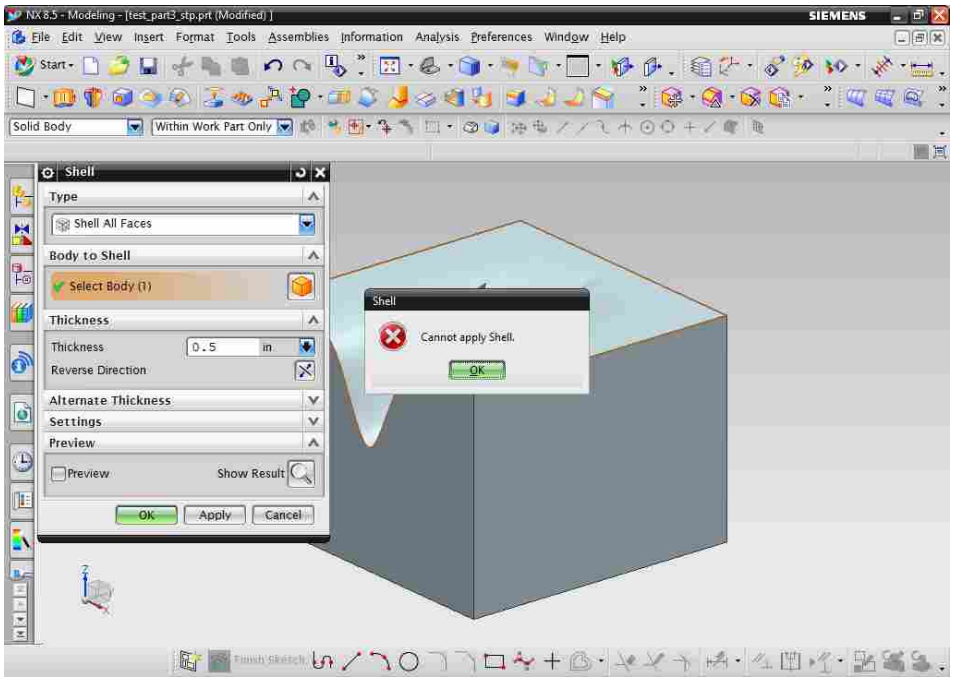


Figure 5-5: NX Part 3

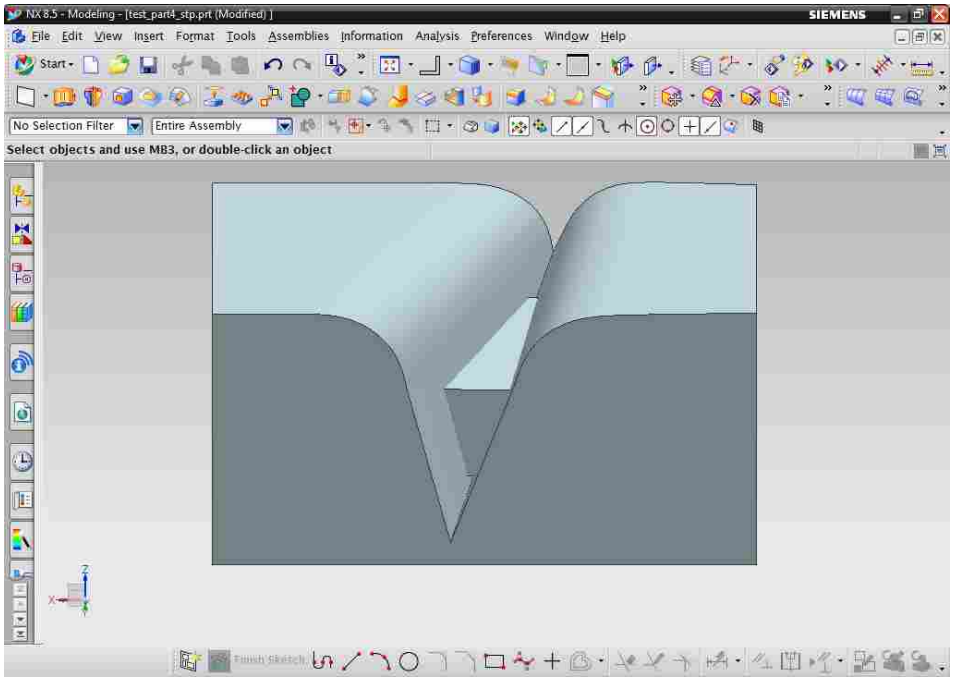


Figure 5-6: NX Part 4

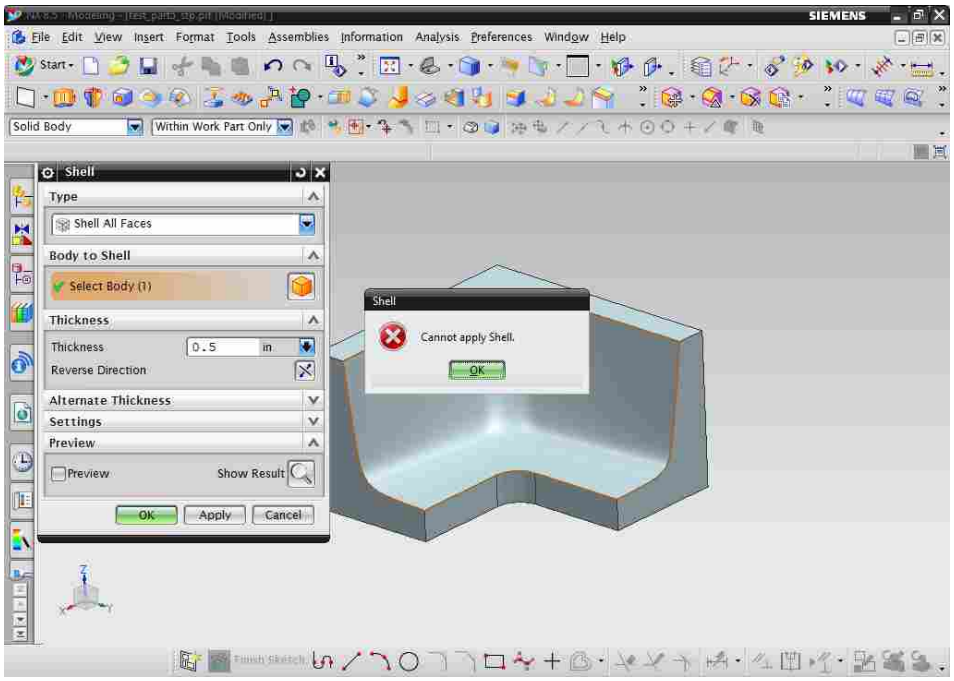


Figure 5-7: NX Part 5

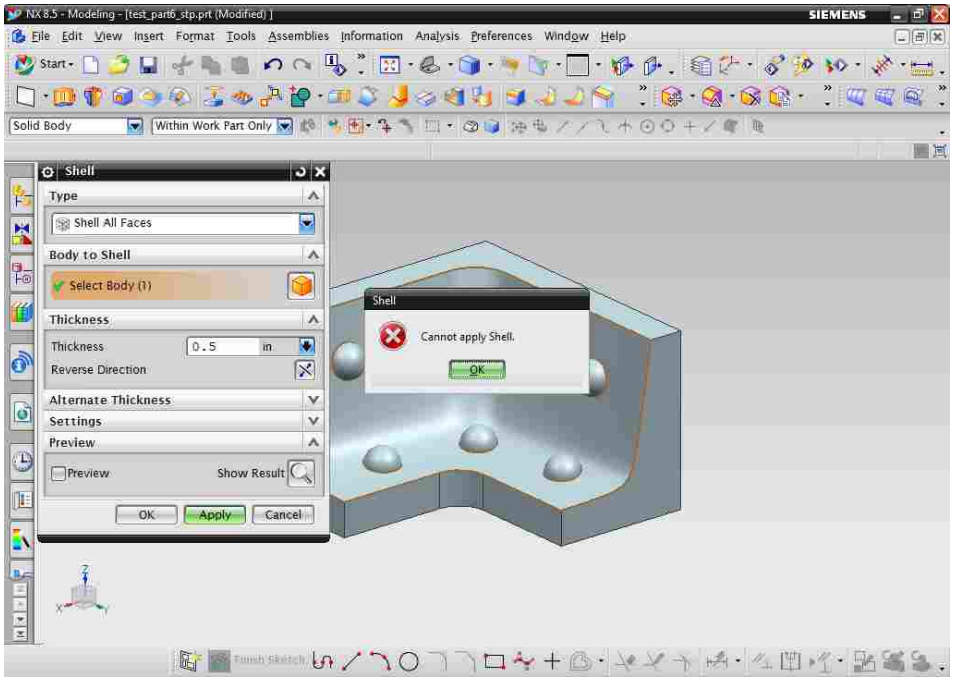


Figure 5-8: NX Part 6

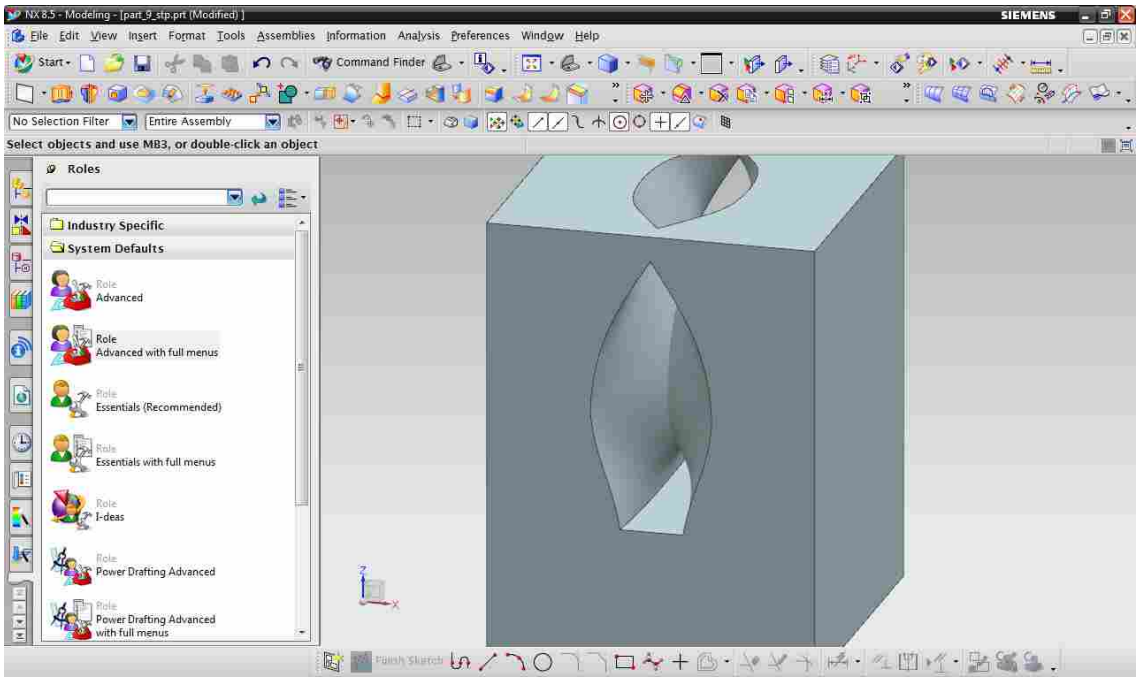


Figure 5-9: NX Part 7

5.2.2 Pro Engineer Results

The following images show the results of offsetting the test parts by way of Pro Engineer. For Pro Engineer, the basic test cases were offset correctly, however, all five complex text cases failed to produce any results for the given offset distance. The complex parts resulted in the Failure Diagnostics Window appearing.

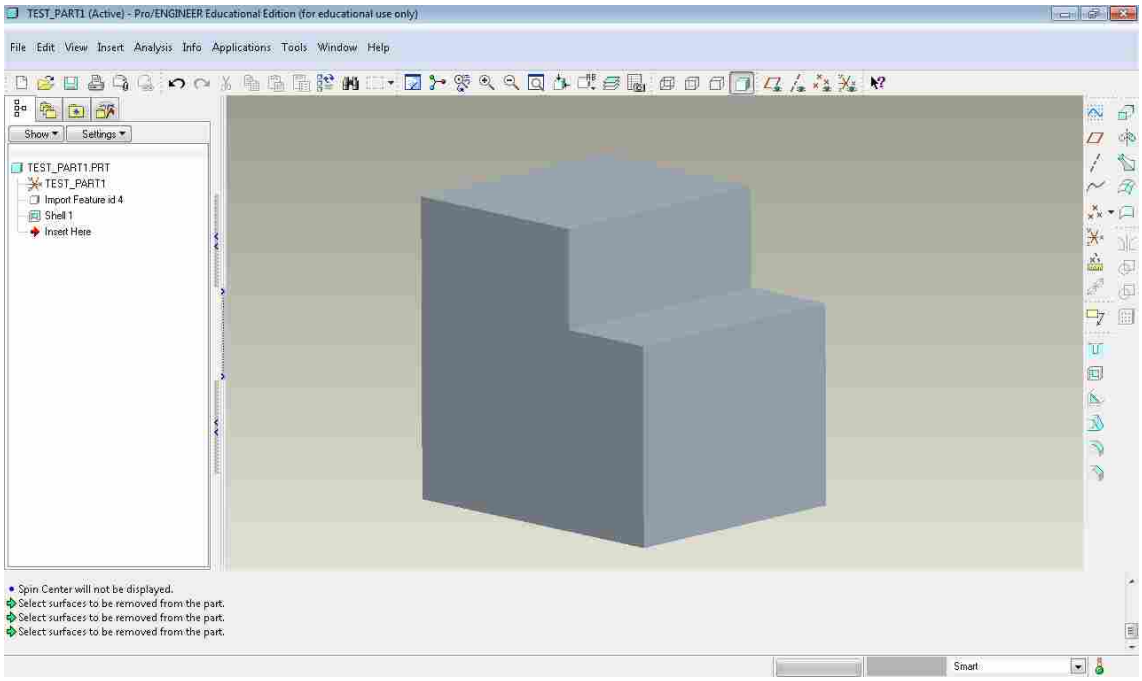


Figure 5-10: Pro/E Part 1

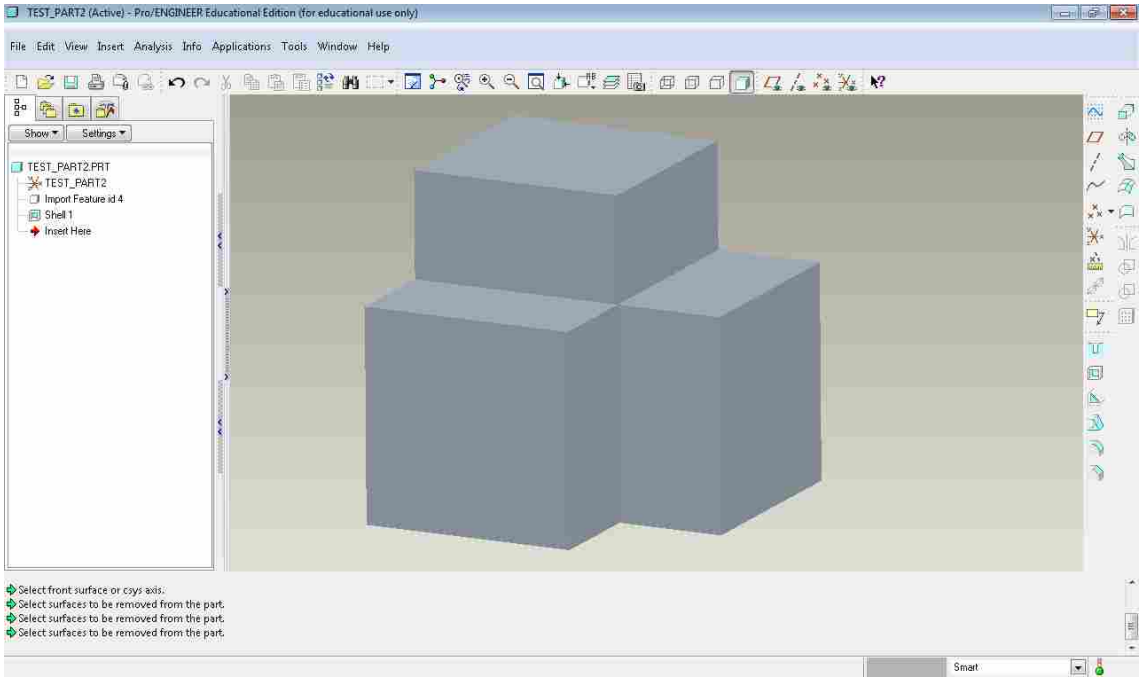


Figure 5-11: Pro/E Part 2

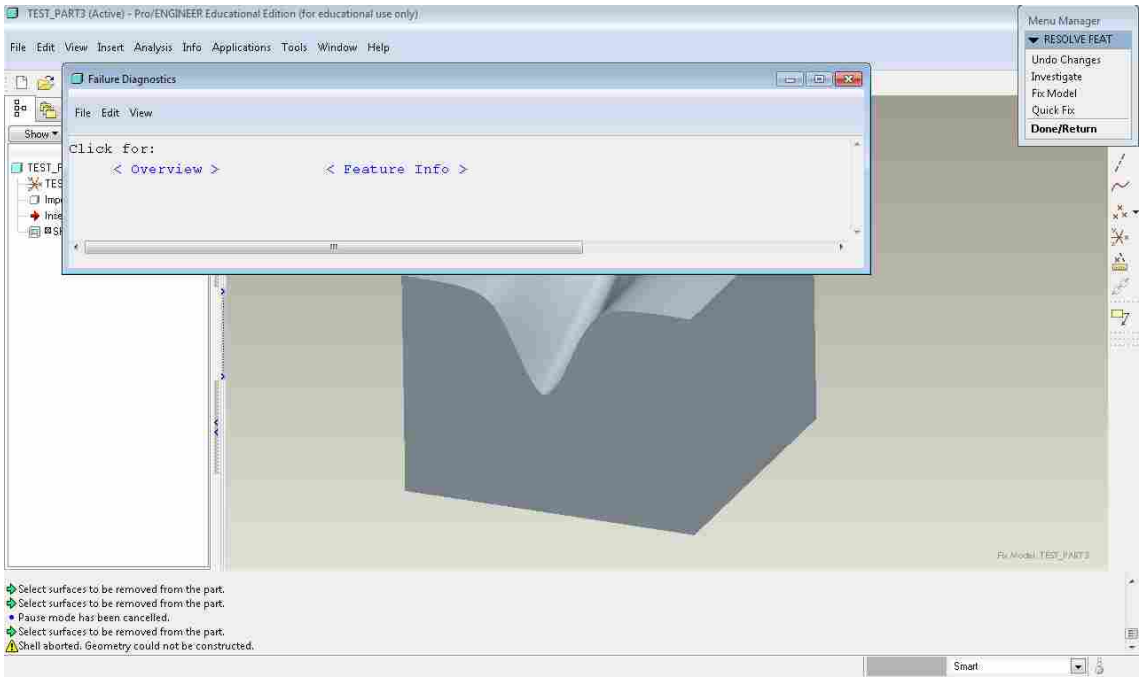


Figure 5-12: Pro/E Part 3

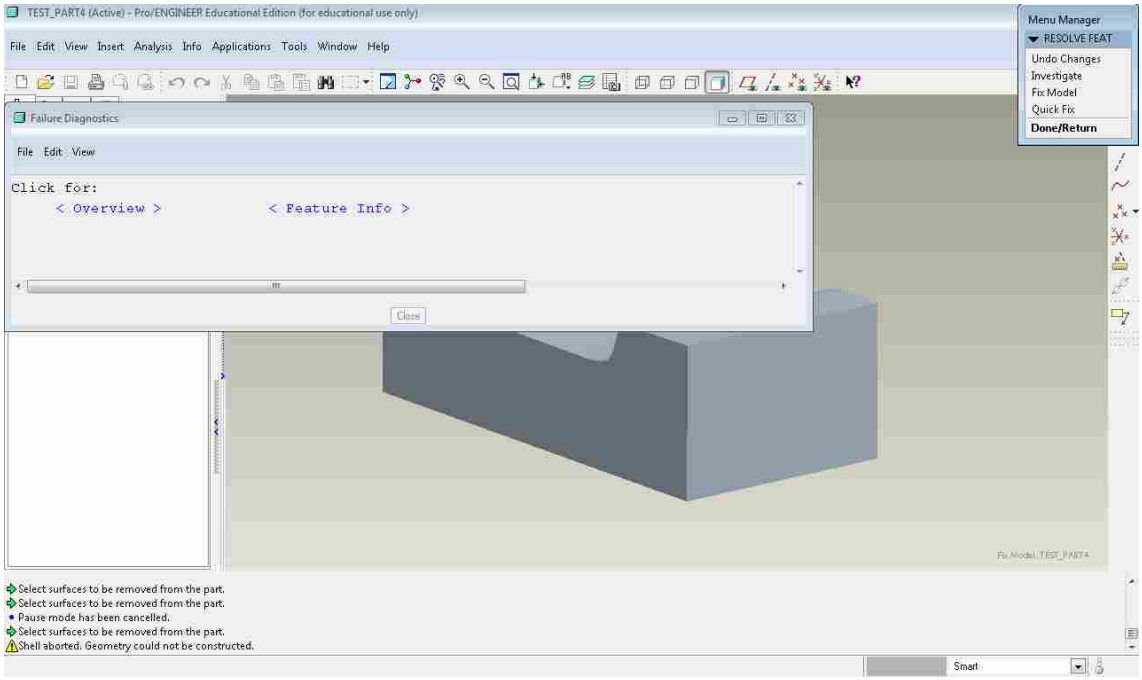


Figure 5-13: Pre/E Part 4

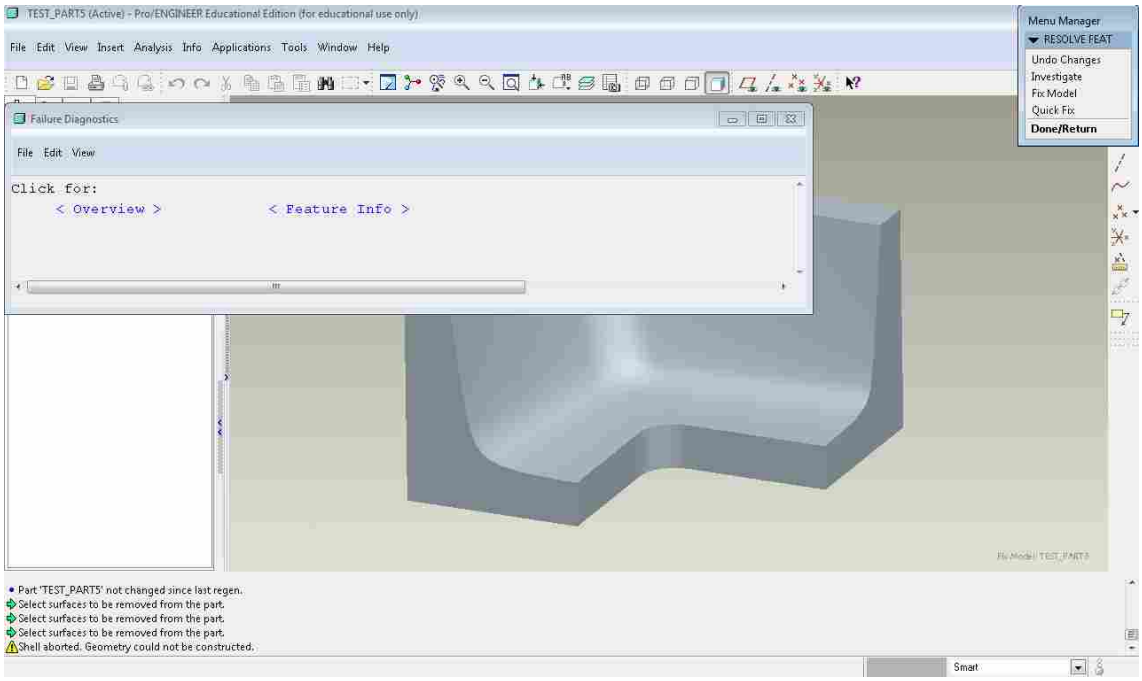


Figure 5-14: Pro/E Part 5

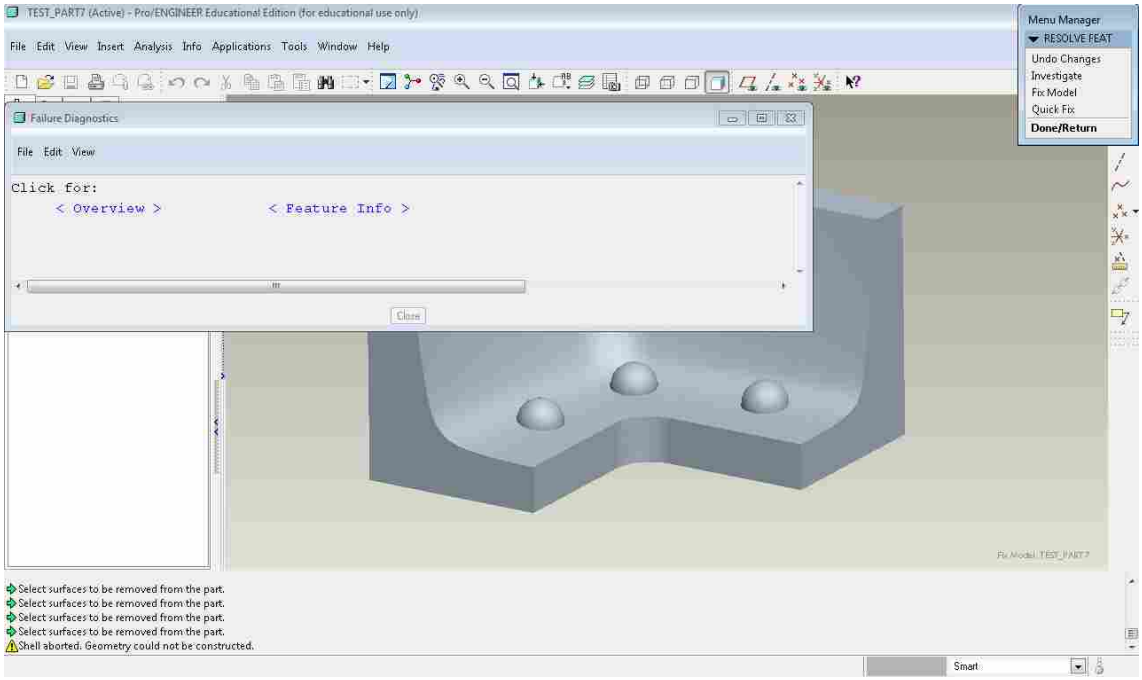


Figure 5-15: Pro/E Part 6

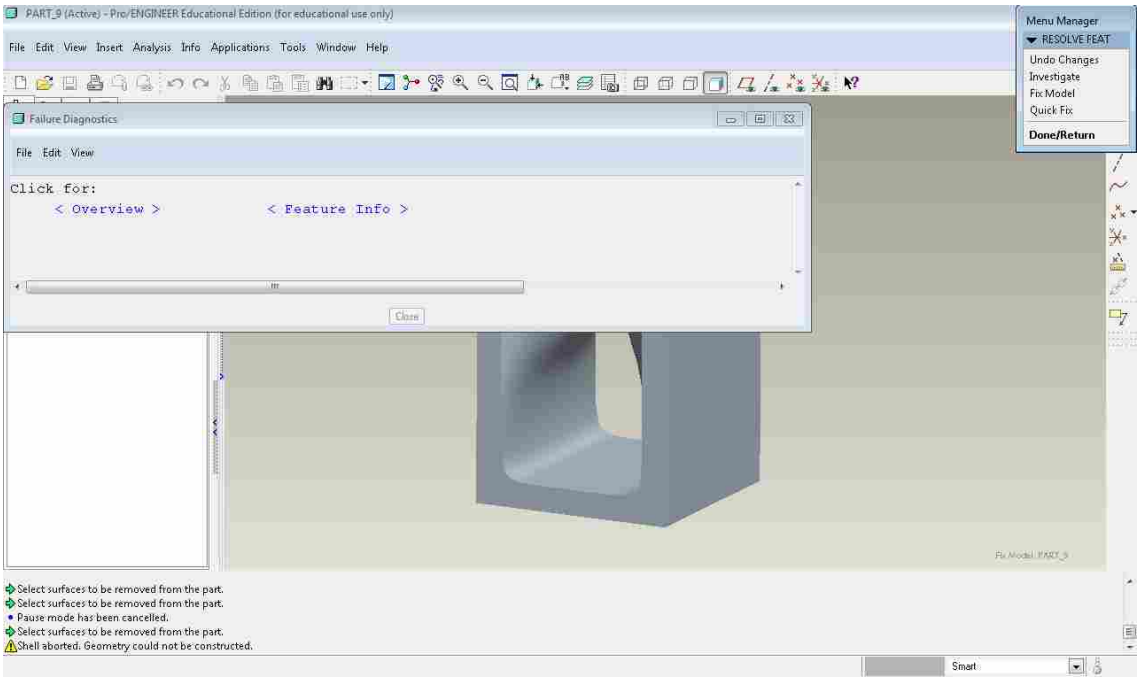


Figure 5-16: Pro/E Part 7

5.2.3 CATIA Results

The following images show the results of offsetting the test parts by way of CATIA. Like Pro Engineer, CATIA also only succeeded on the basic test cases. All other test cases resulted in an error message that read, “Current offset value leads to a local degeneration on a surface.”

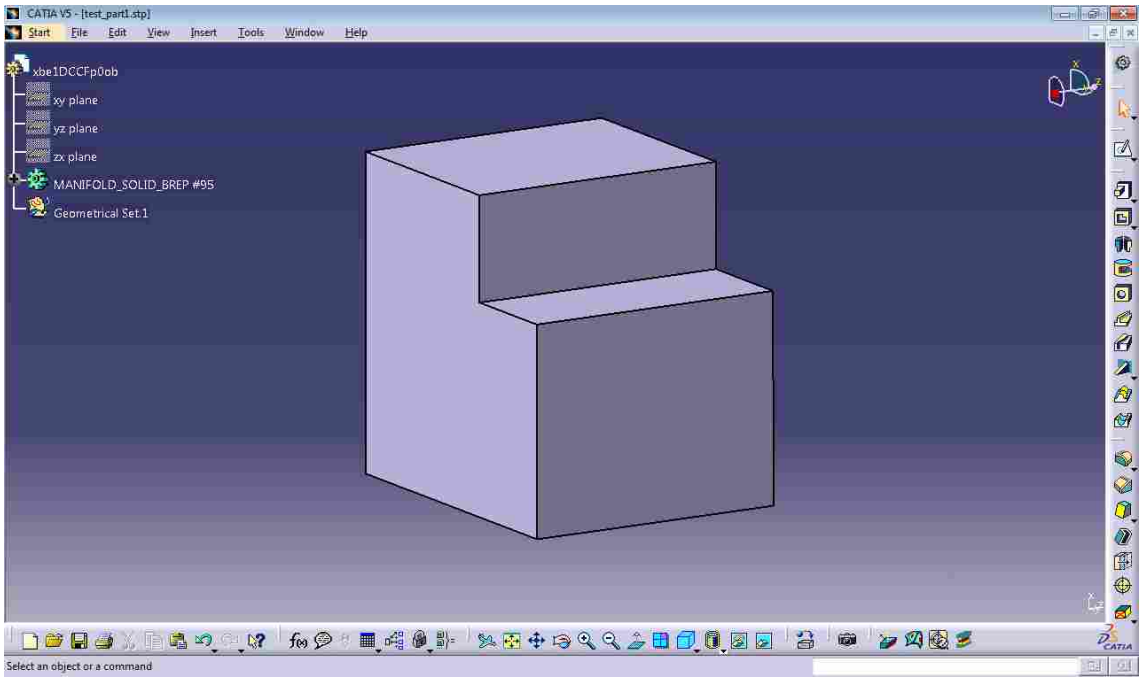


Figure 5-17: CATIA Part 1

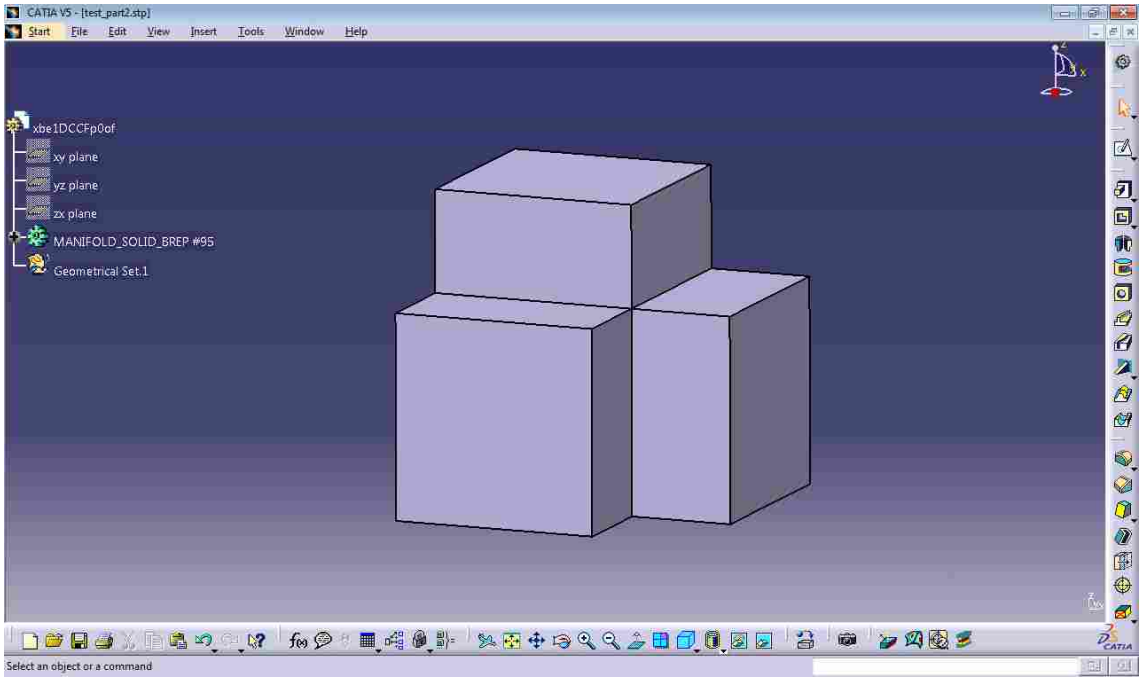


Figure 5-18: CATIA Part 2

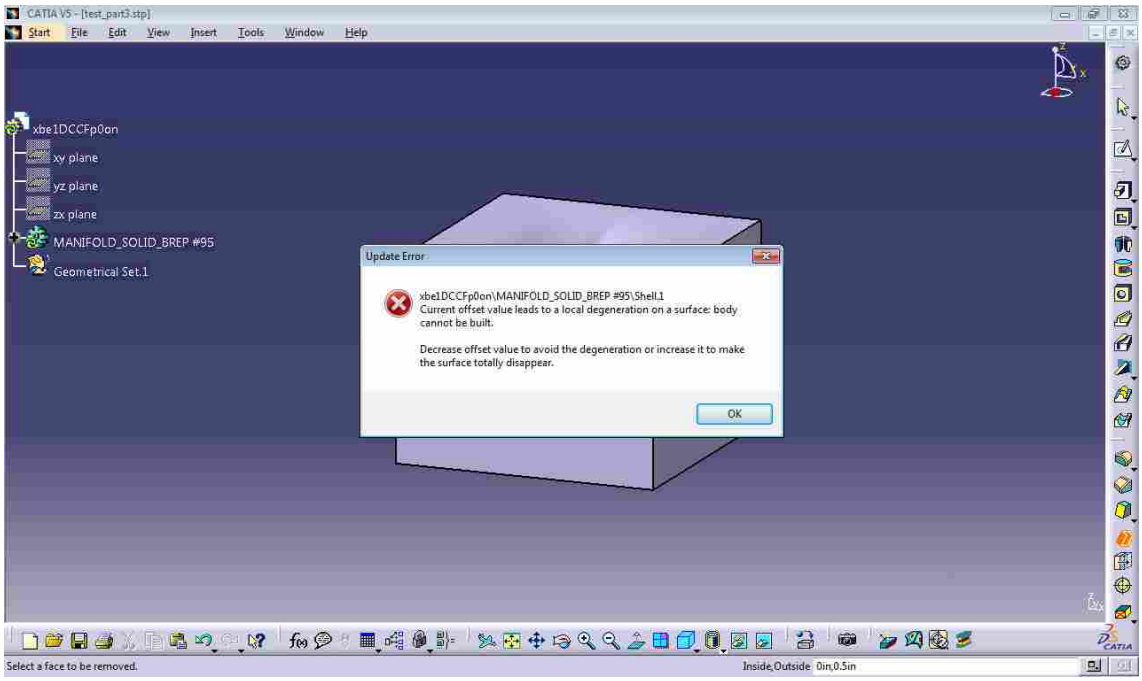


Figure 5-19: CATIA Part 3

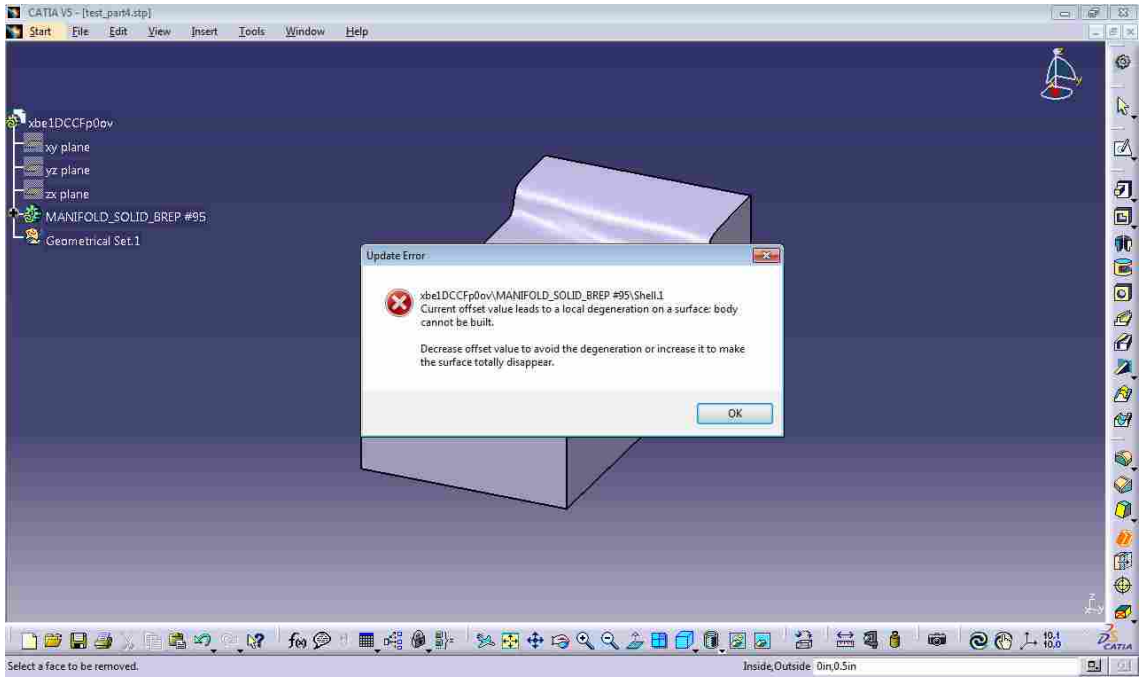


Figure 5-20: CATIA Part 4

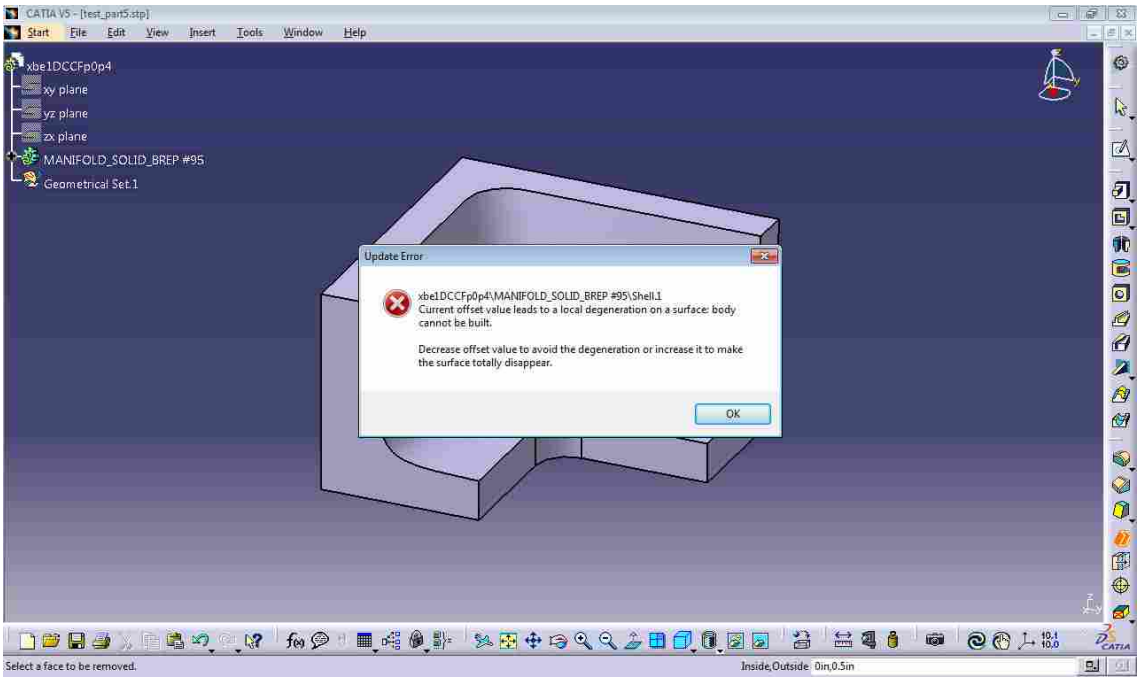


Figure 5-21: CATIA Part 5

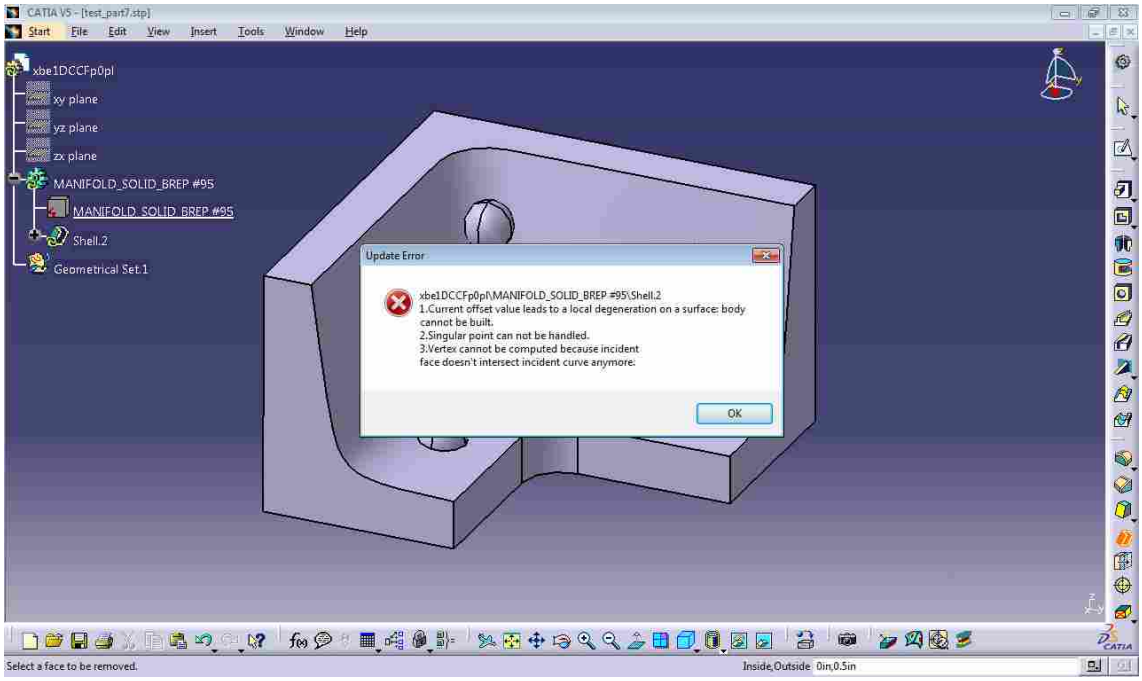


Figure 5-22: CATIA Part 6

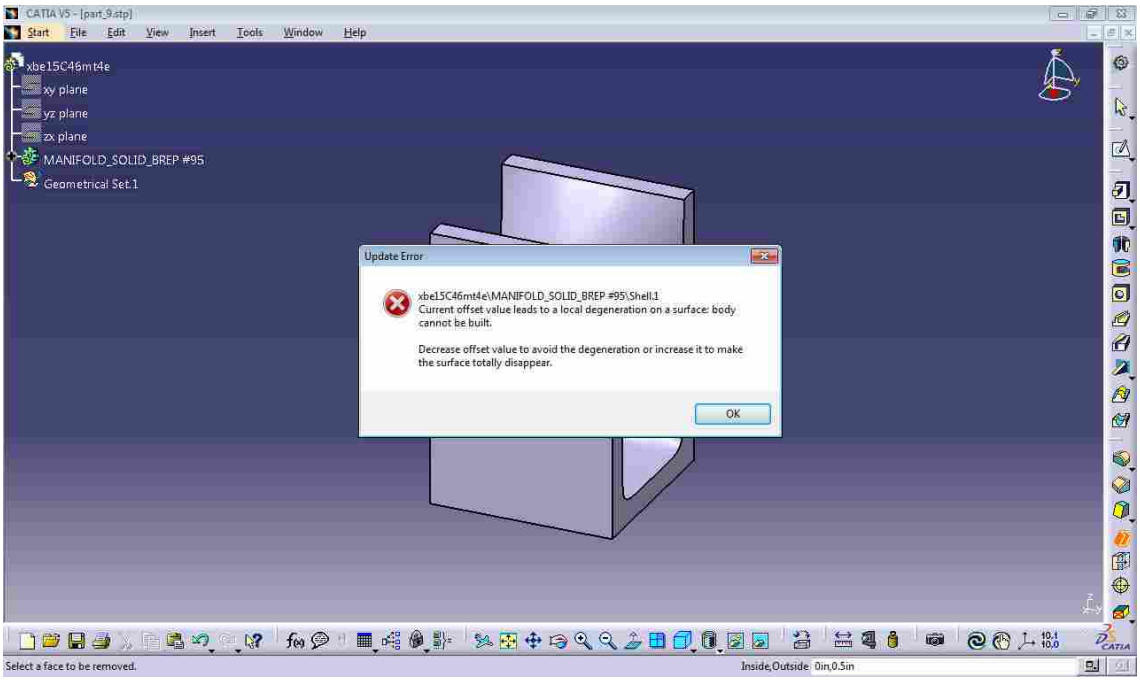


Figure 5-23: CATIA Part 7

5.2.4 SolidWorks Results

The following images show the results of offsetting the test parts by way of SolidWorks. SolidWorks was able to offset not only the basic parts but also a few of the complex parts as well. All five of the complex parts produced a result but three of them are very poor approximations. Part 3 resulted in a very noisy surface where the degeneracy was removed. Part 4 produced an offset similar to NX, the offset surface is produced but it folds back and intersects the original part which is undesirable. Part 6 also produced a result but completely removed the free-form surface resulting in a simple block that does not accurately reflect the true offset surface.

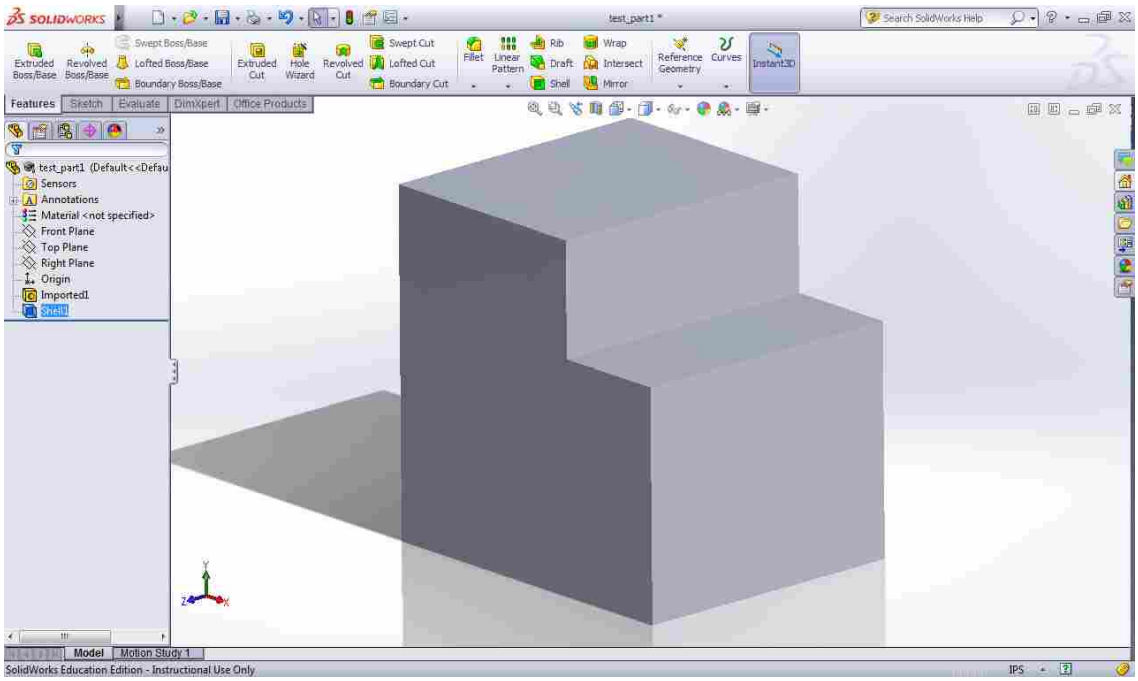


Figure 5-24: SolidWorks Part 1

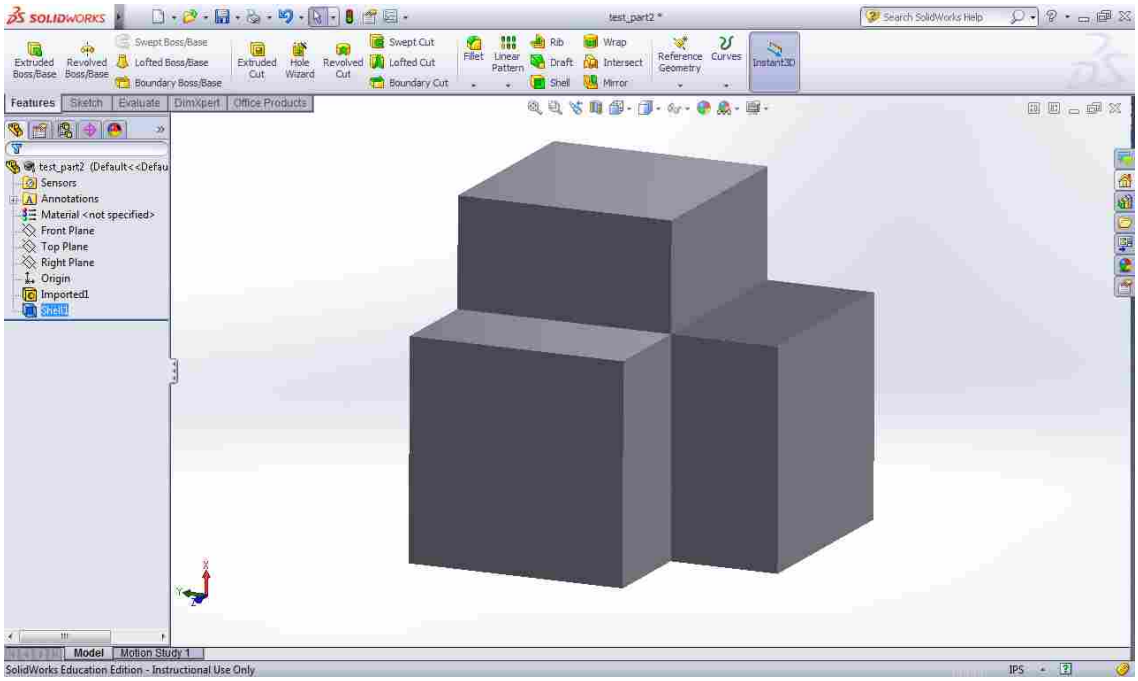


Figure 5-25: SolidWorks Part 2

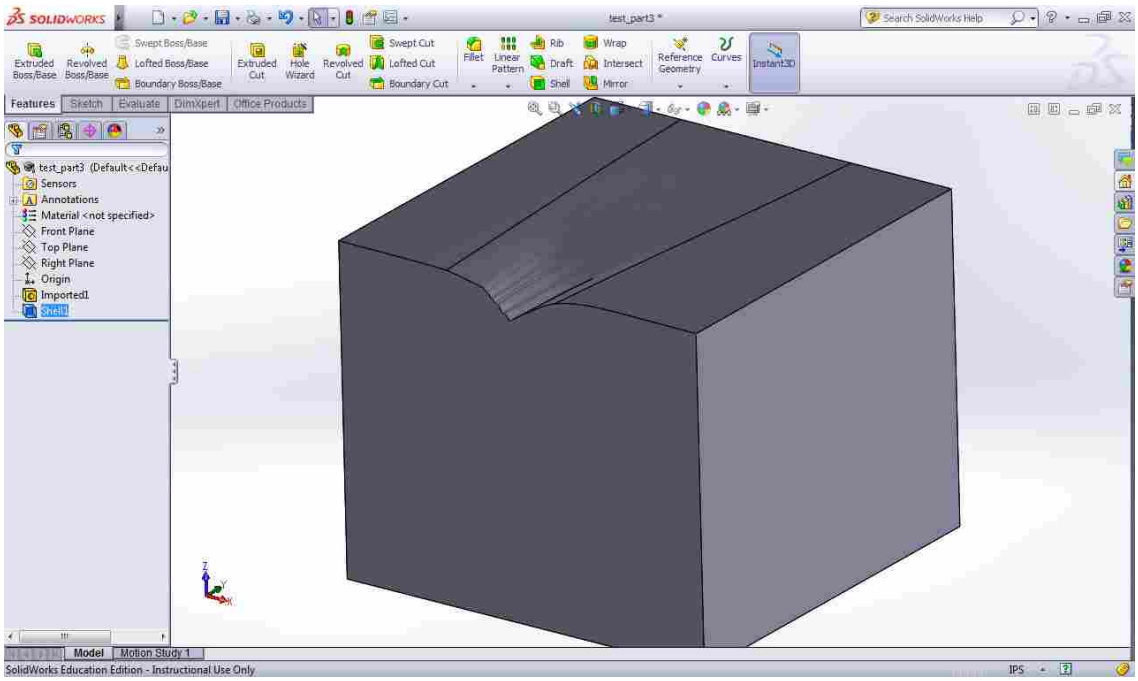


Figure 5-26: SolidWorks Part 3

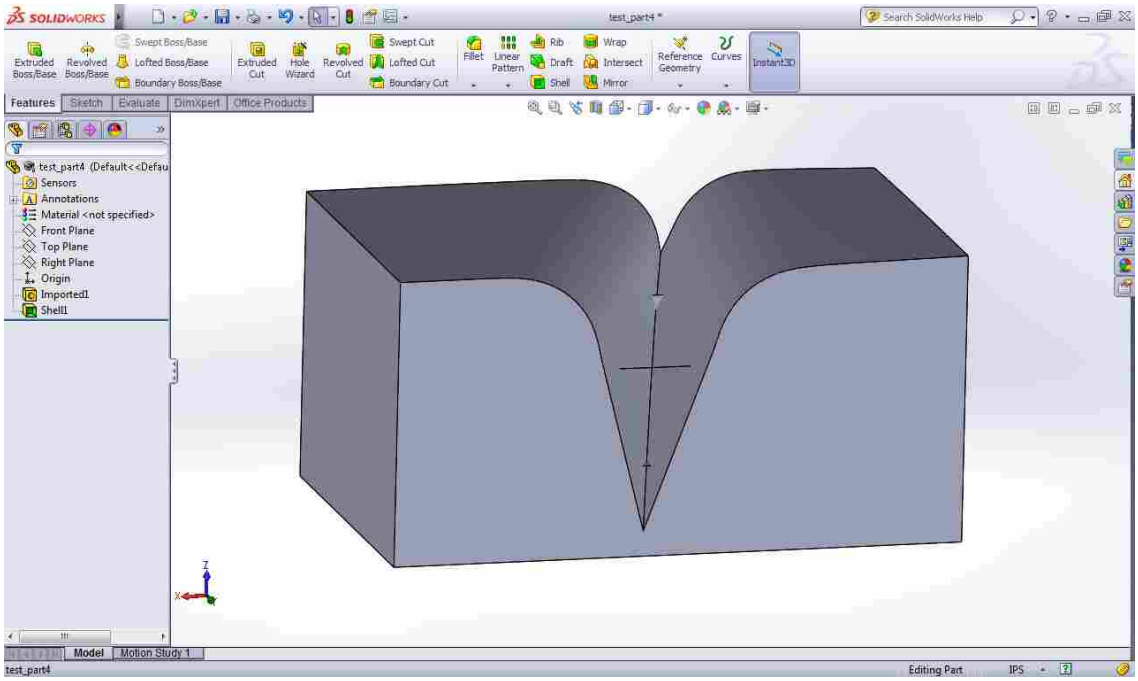


Figure 5-27: SolidWorks Part 4

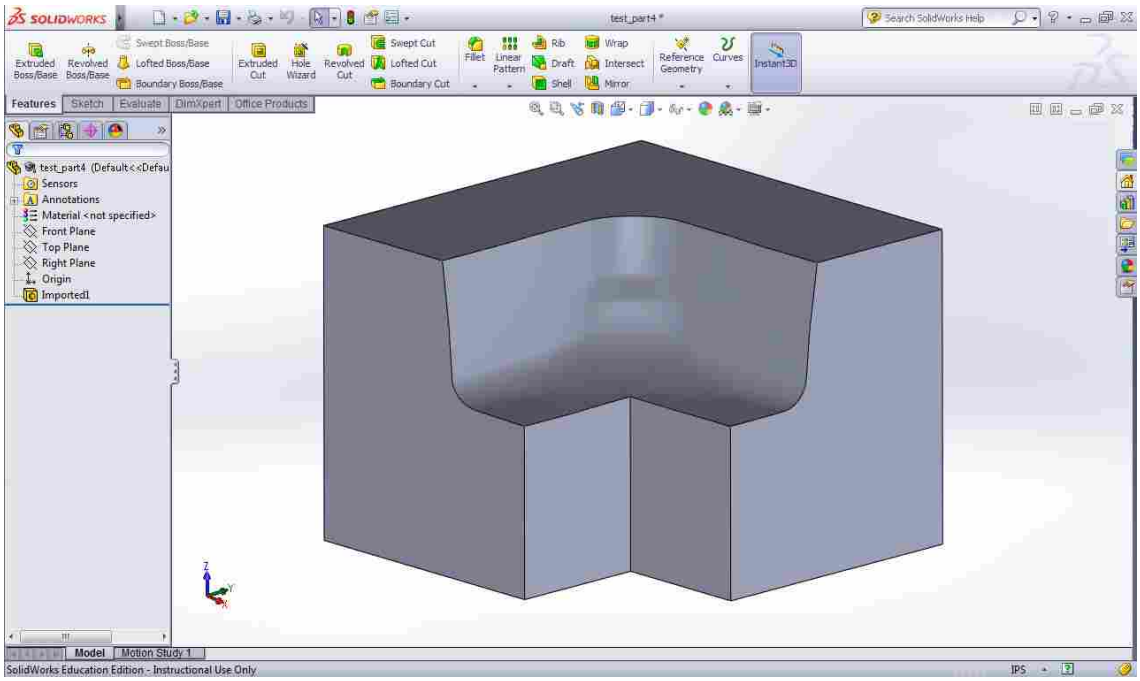


Figure 5-28: SolidWorks Part 5

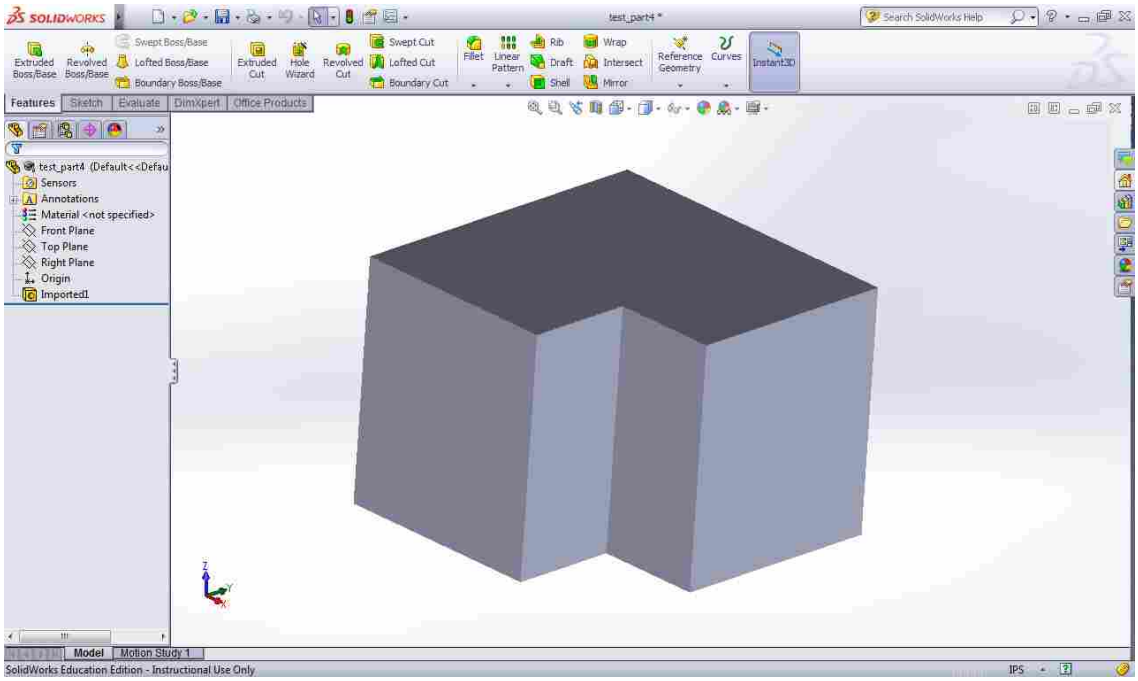


Figure 5-29: SolidWorks Part 6

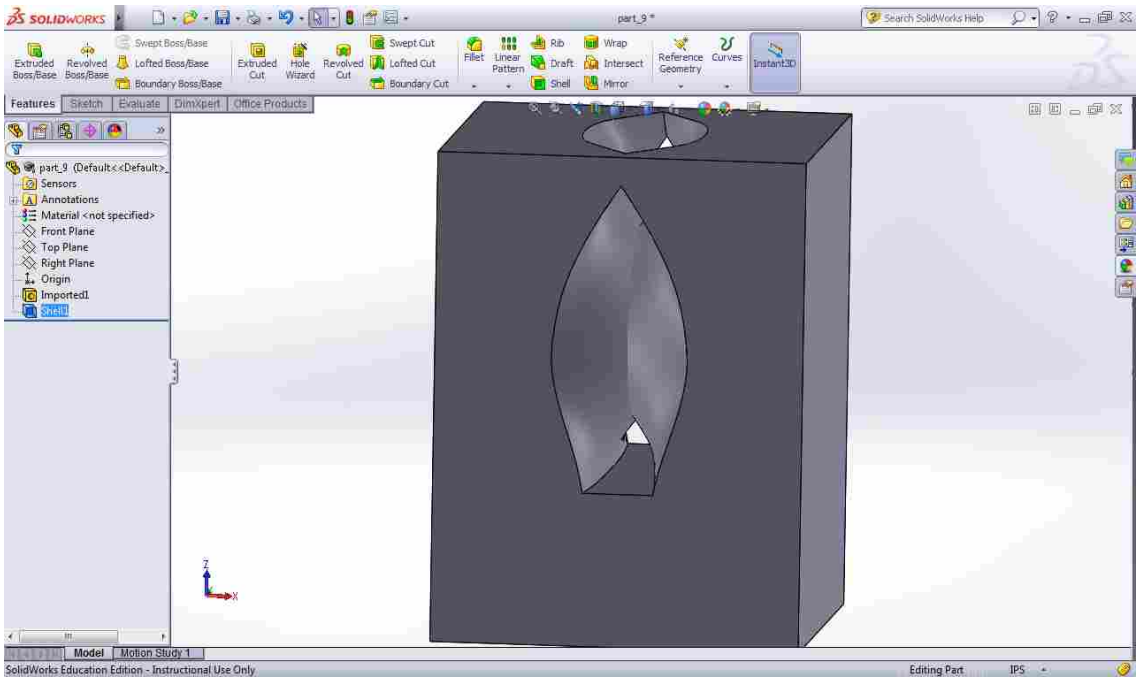


Figure 5-30: SolidWorks Part 7

5.2.5 Inventor Results

The following images show the results of offsetting the test parts by way of Inventor. Inventor was the most successful CAD system for offsetting parts 3 and 4. Inventor correctly removed the self-intersections and in the case of part 3 more accurately approximated the offset surface than SolidWorks. SolidWorks was the only other CAD system that produced an offset surface for part 3 but the approximation was very poor.

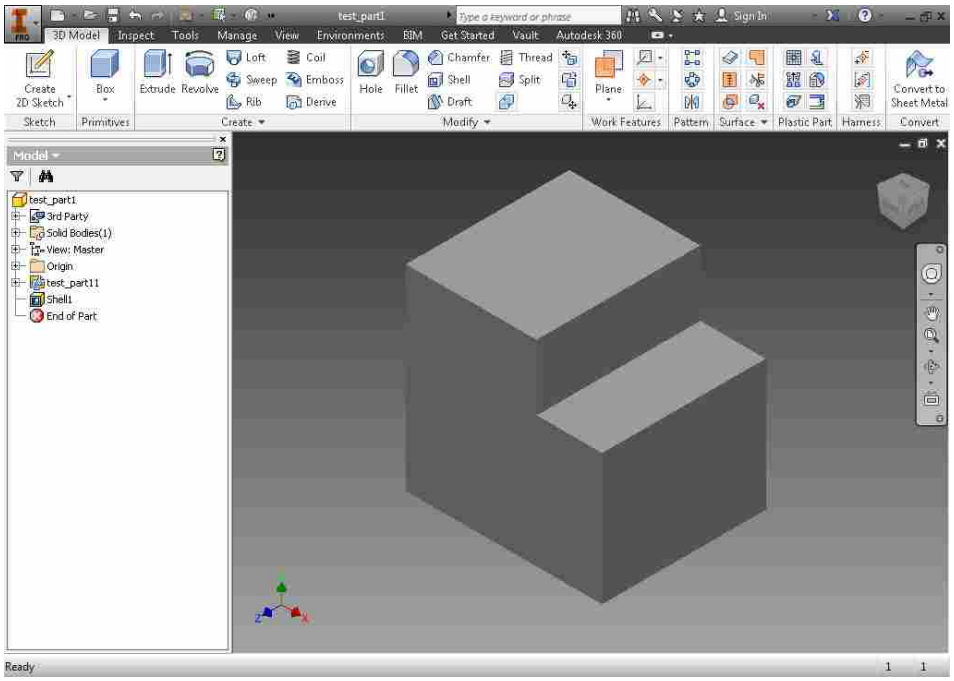


Figure 5-31: Inventor Part 1

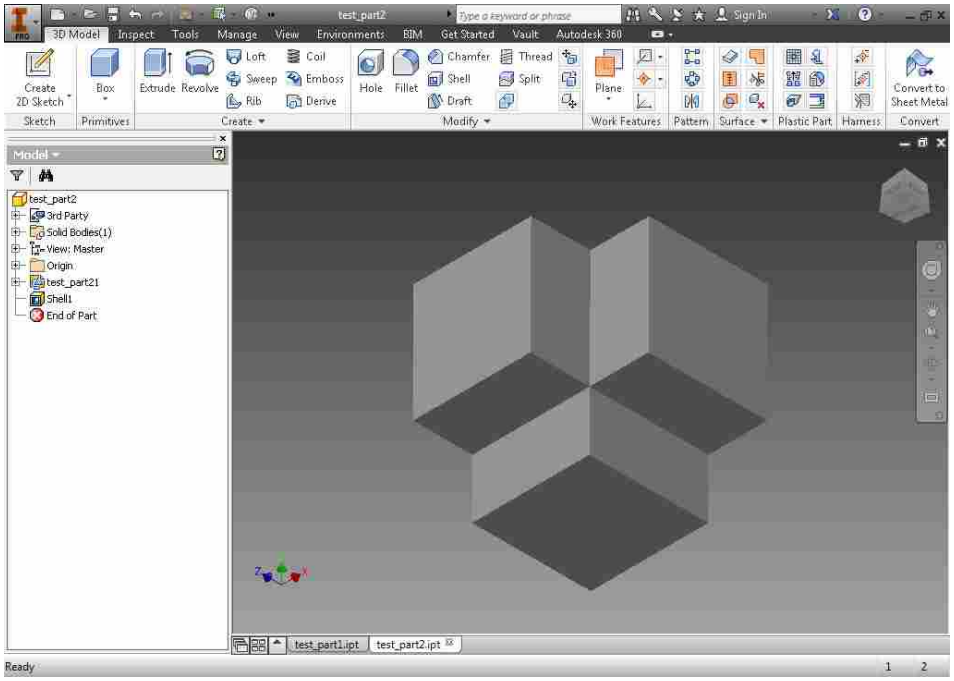


Figure 5-32: Inventor Part 2

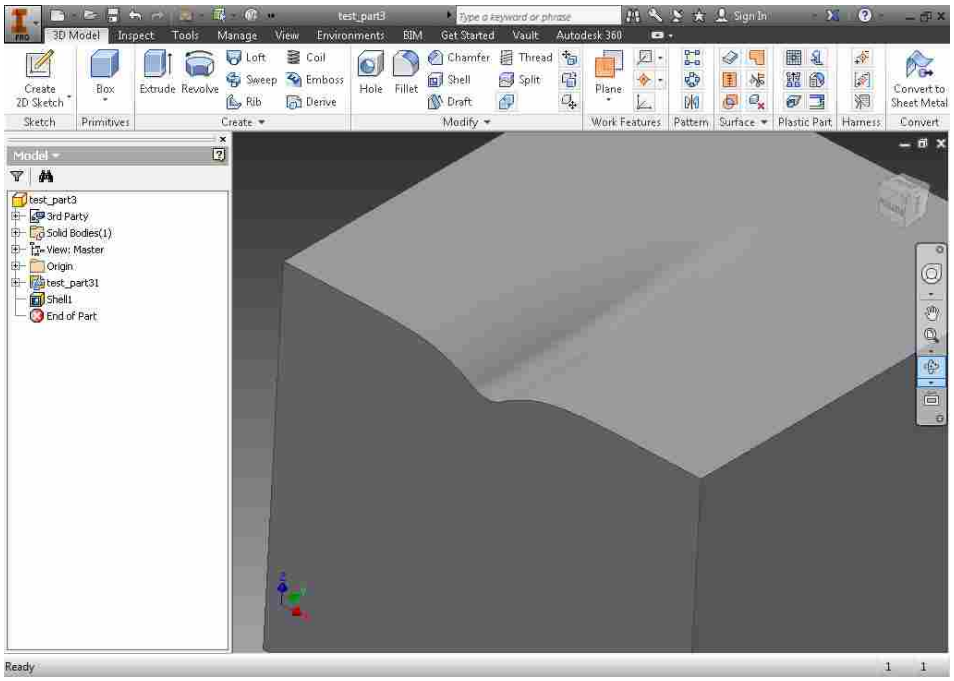


Figure 5-33: Inventor Part 3

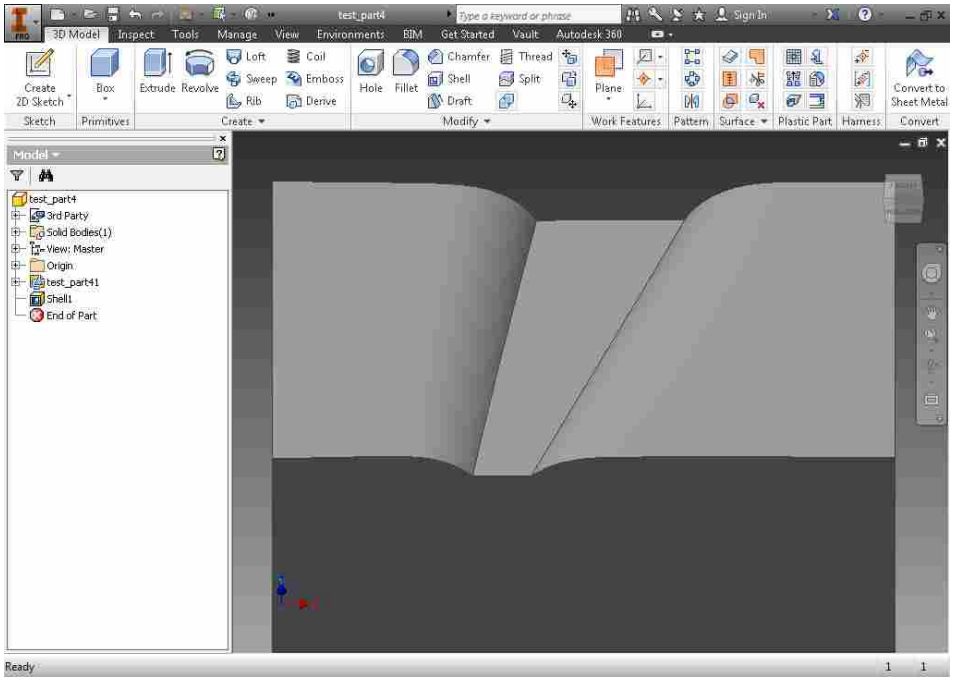


Figure 5-34: Inventor Part 4

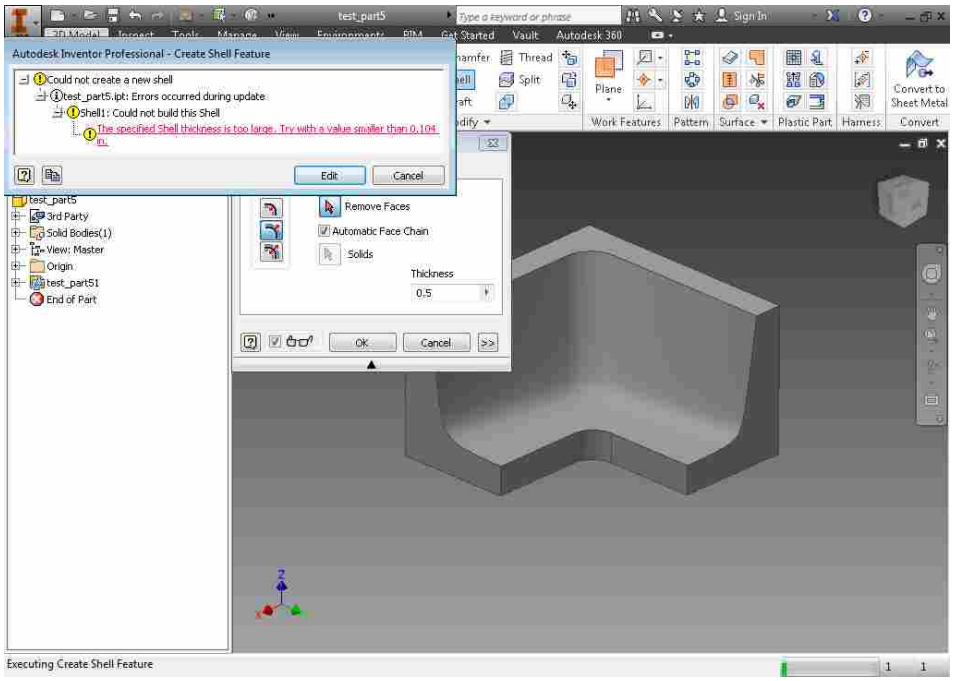


Figure 5-35: Inventor Part 5

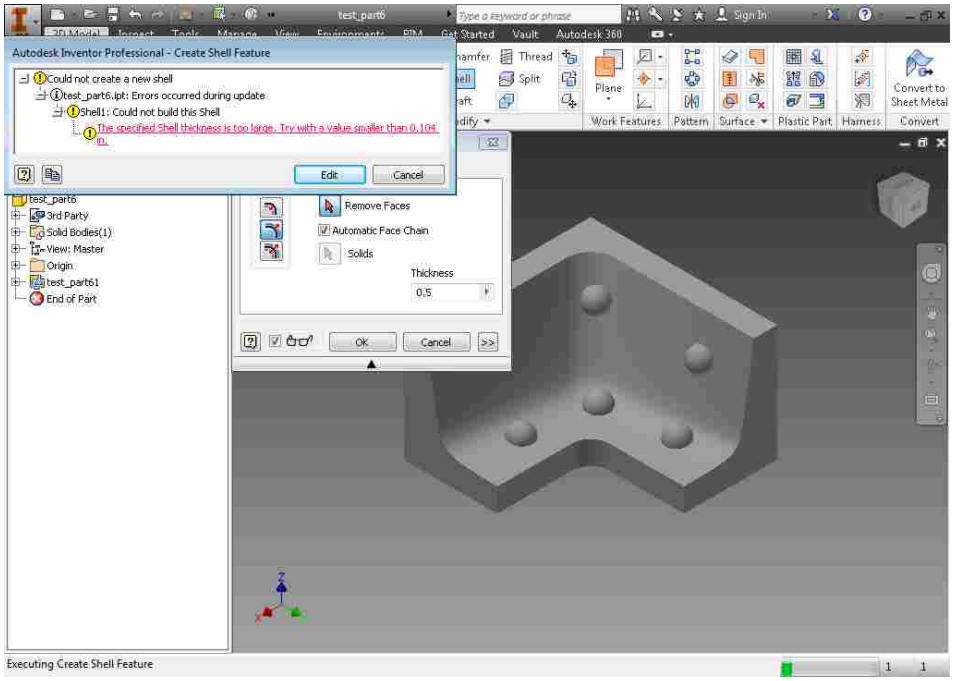


Figure 5-36: Inventor Part 6

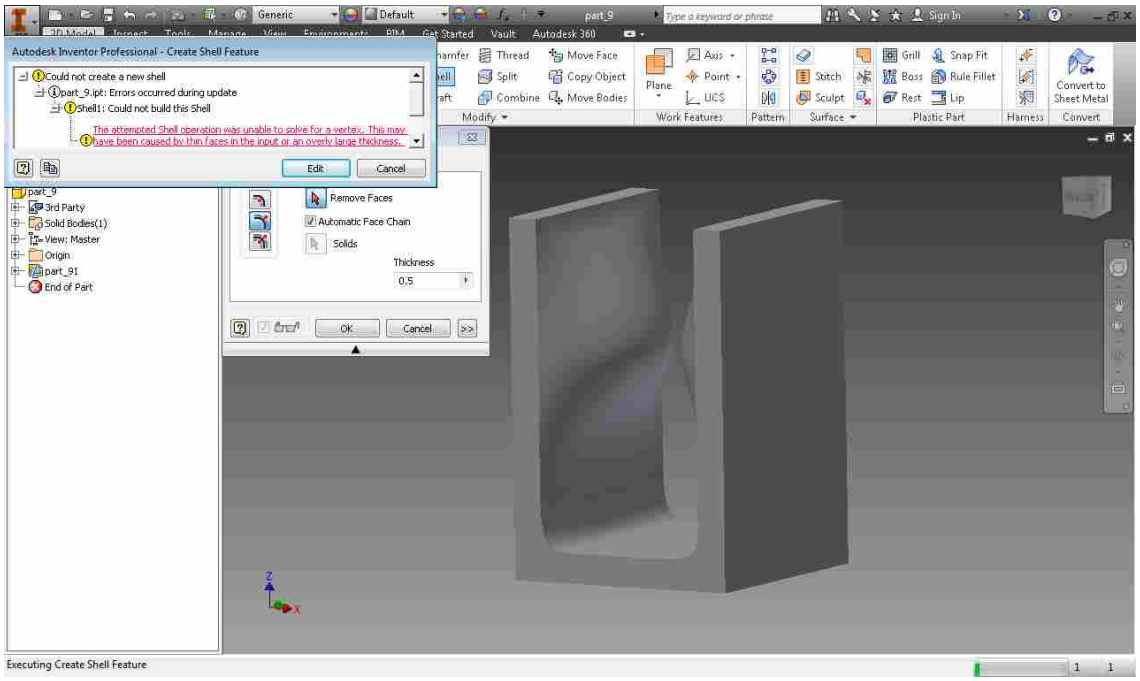


Figure 5-37: Inventor Part 7

5.3 Summary of CAD System Results

In order to compare the results of the CAD systems' offset surfaces, a measure of accuracy is used. For many of the test cases, no offset was produced, therefore, no measurement can be taken. For test cases where results were produced, the free-form surface of the test part is examined. The simple faces are not measured because they were offset directly and therefore have no error. The free-form surfaces are measured for accuracy by choosing a sufficiently large number of points on the surface and calculating their minimum distance back to the original surface. The distance for each point should in theory always be equal to the offset distance. However, there is some variability in these distances. The following table summarizes the accuracy of the offset surfaces by comparing their maximum and average errors.

Table 5-1: Accuracy of CAD Results

CAD System	Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7
NX	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	NA	Max: % 100.0 Avg: % 46.3	NA	NA	Max: % 0.0 Avg: % 0.0
Pro/E	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	NA	NA	NA	NA	NA
CATIA	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	NA	NA	NA	NA	NA
SolidWorks	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	Max: % 34.8 Avg: % 3.5	Max: % 100.0 Avg: % 46.3	Max: % 39.5 Avg: % 9.1	NA	Max: % 0.0 Avg: % 0.0
Inventor	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	Max: % 17.7 Avg: % 1.7	Max: % 3.3 Avg: % 0.0	NA	NA	NA

Table 5.1 show that some of the CAD systems were able to produce very accurate offsets for some of the test cases. Many of the tests, however, could not be offset at all. In some cases an offset was produced but was very inaccurate.

5.4 Offset Tool Results

The following images display the results of the hybrid method. The results are different from the CAD systems, instead of extending and trimming the offset faces the hybrid method naturally creates rounded corners between faces. The results also show that the self-intersections and changes in topology were handled correctly. These results demonstrate the value of this new process for creating offset surfaces of complex CAD geometry.

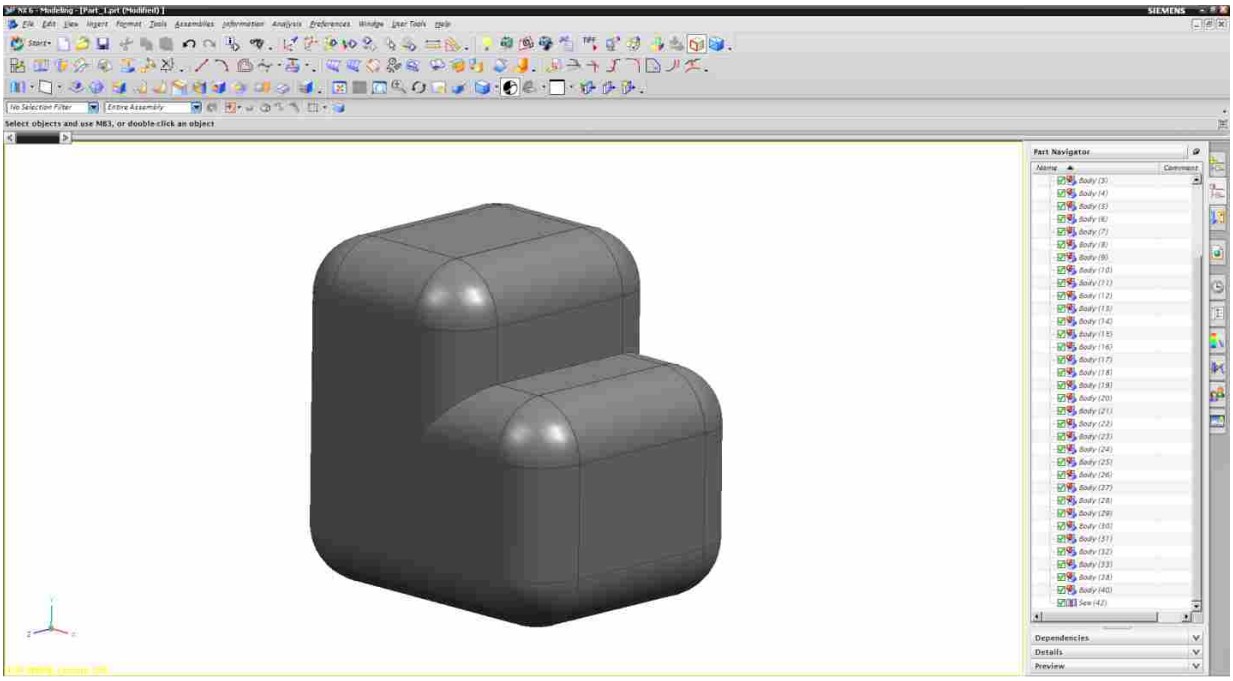


Figure 5-38: Hybrid Method Part 1

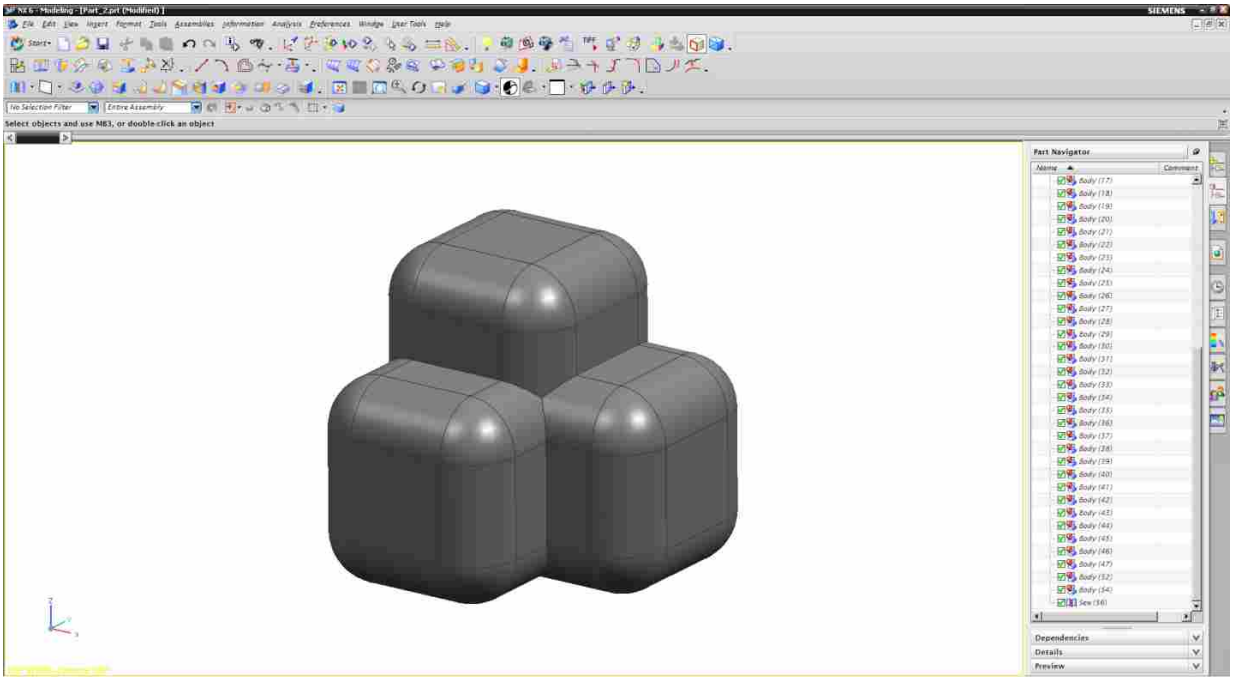


Figure 5-39: Hybrid Method Part 2

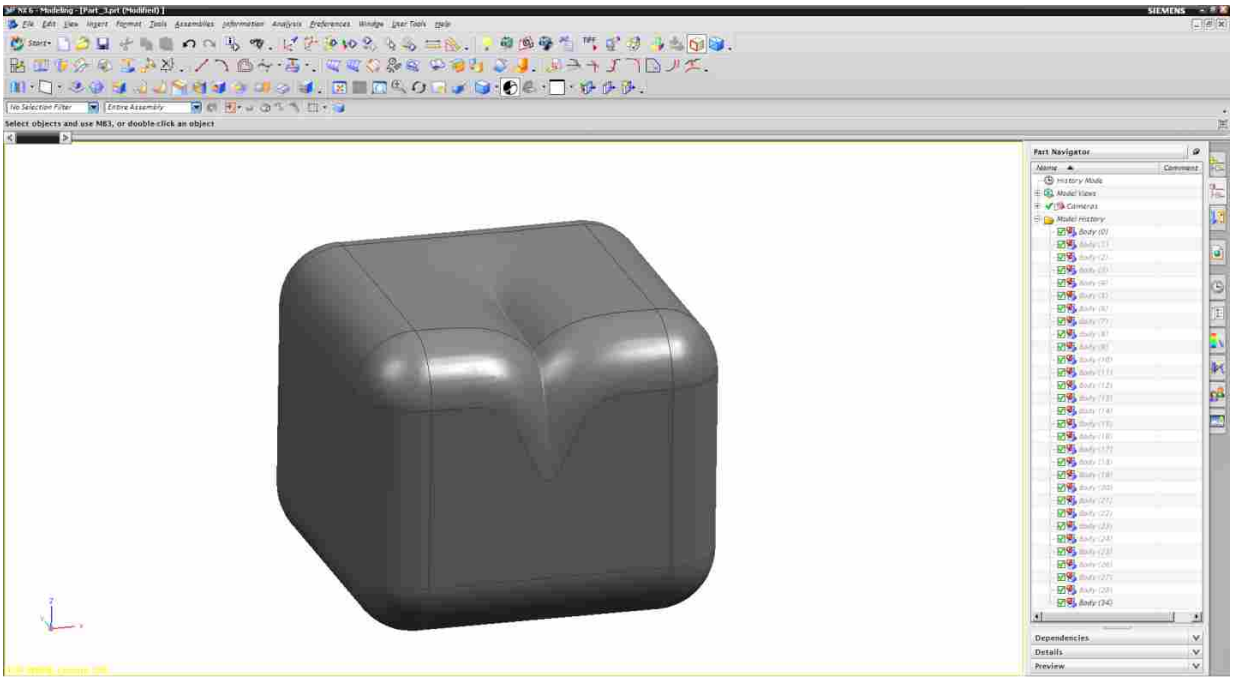


Figure 5-40: Hybrid Method Part 3

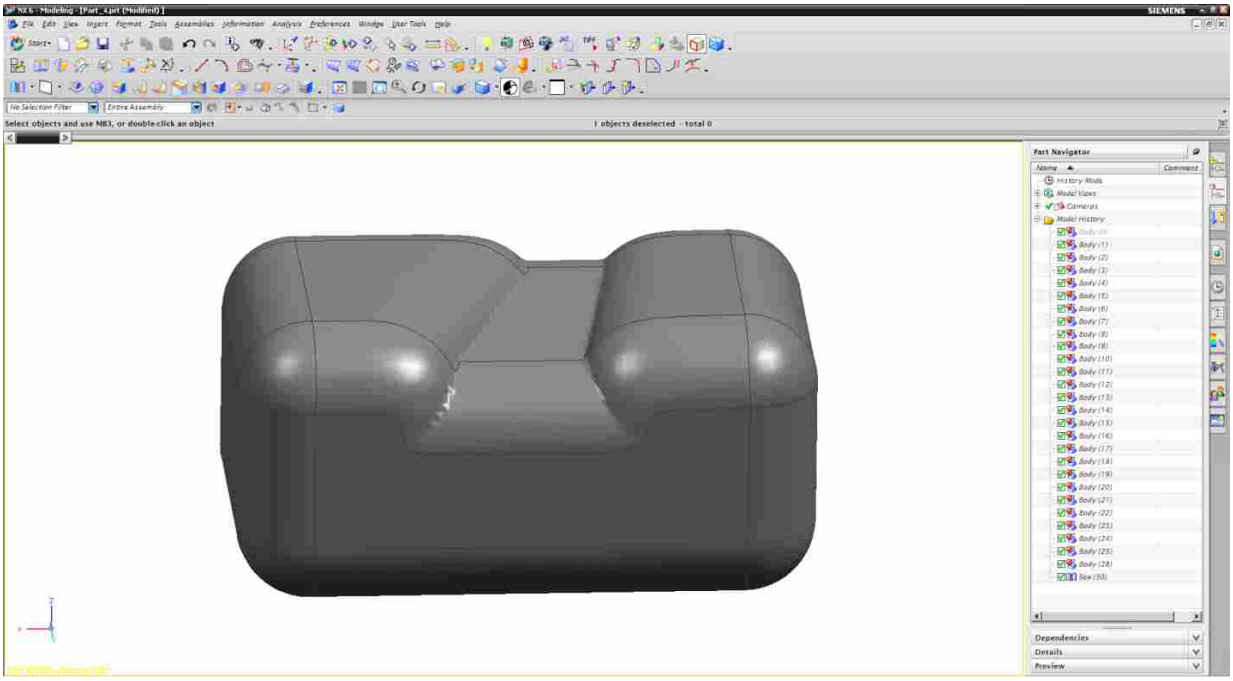


Figure 5-41: Hybrid Method Part 4

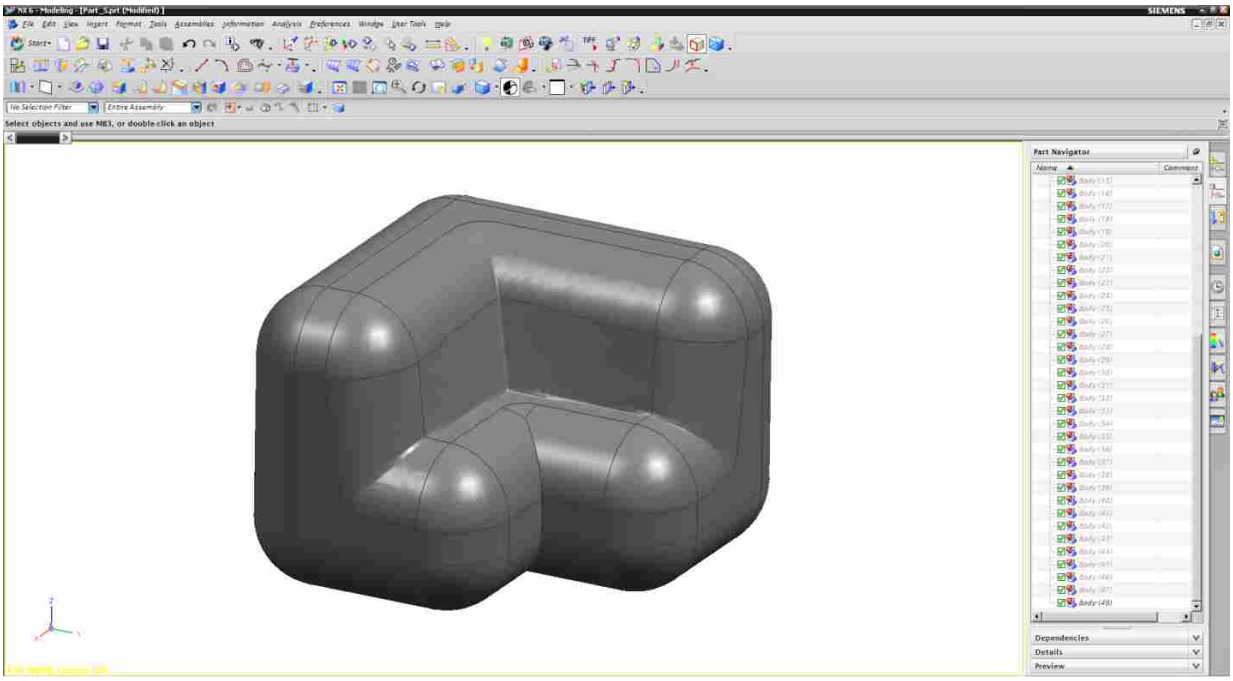


Figure 5-42: Hybrid Method Part 5

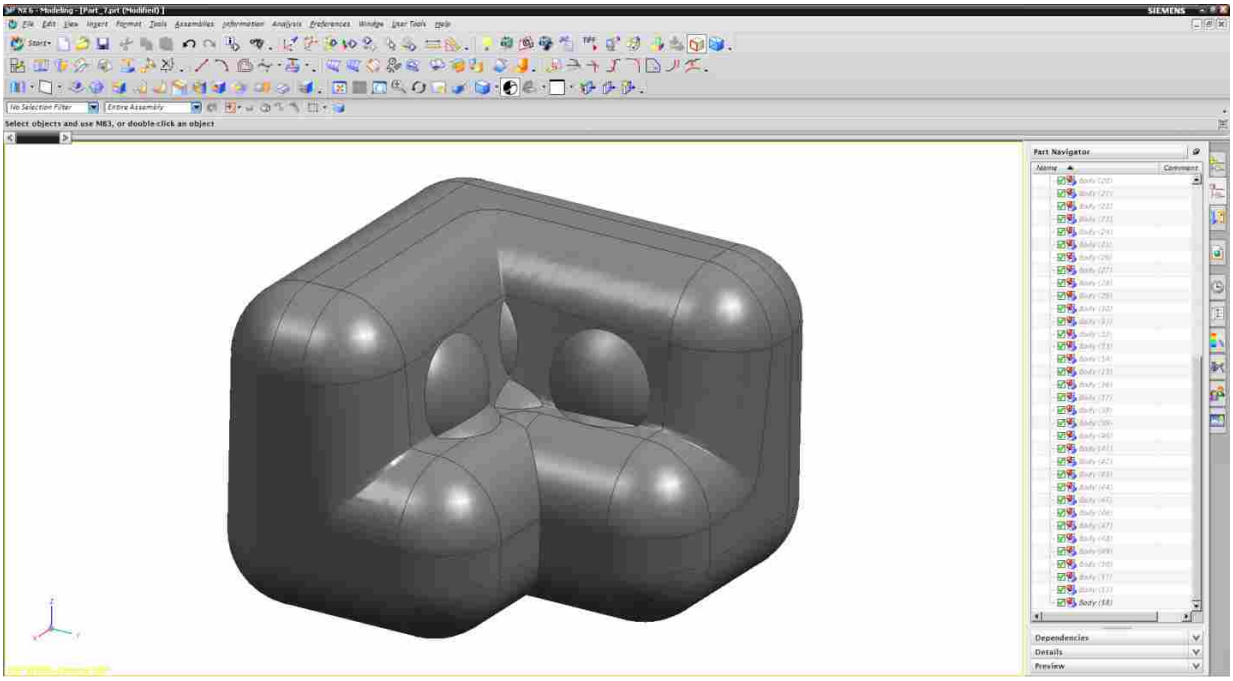


Figure 5-43: Hybrid Method Part 6

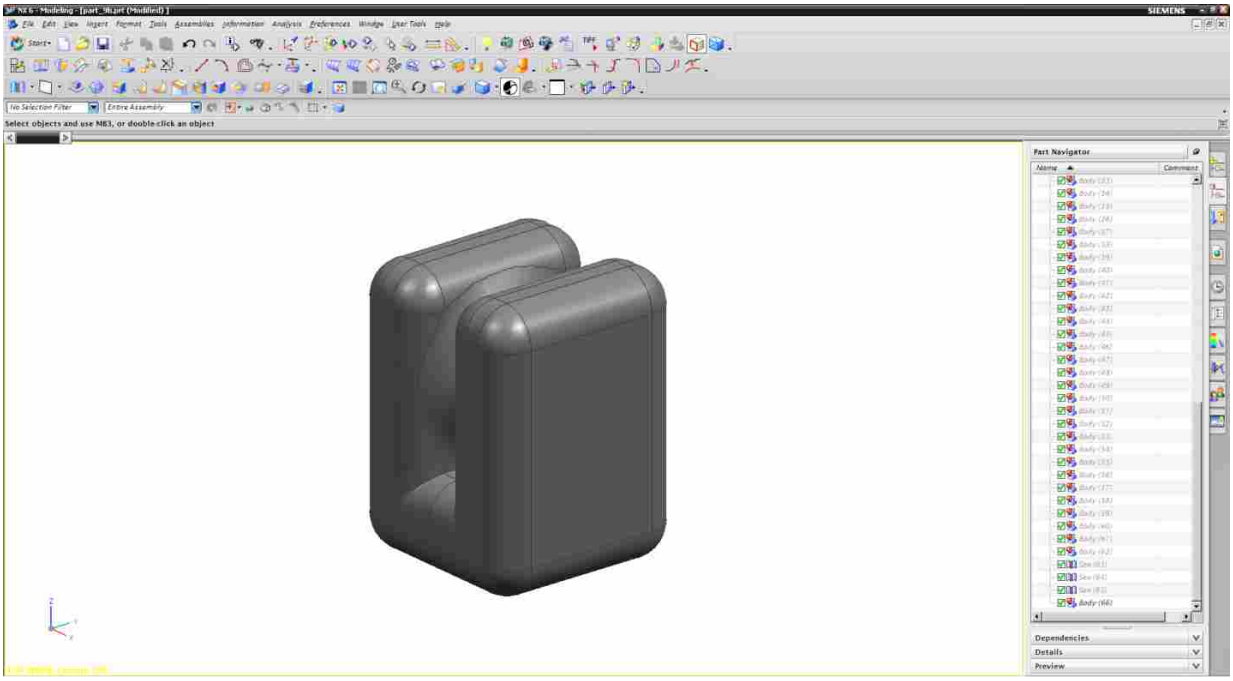


Figure 5-44: Hybrid Method Part 7

5.5 Summary of Hybrid Method Results

As can be seen from the above images, the hybrid method was successful in producing a result for each of the seven test cases. The following table summarizes both the CAD system results and also adds the results of the Hybrid Method. Each of the test cases was run with a grid resolution of 0.01, interestling enough many of the results are more accurate than the grid resolution. As can be seen from table 5.2, the hybrid method produces results that are more accurate that the CAD systems in many of the cases. In cases 3 and 5 the hybrid method produces results that are more accurate those produced by SolidWorks or Inventor. In case 4 the hybrid method produces an accurate offset surface but Inventor's results were better. In case 6 the hybrid method was the only method to return a measurable result. NX and SolidWorks both produced more accurate results for case 7 but again, the hybrid method produced results that are accurate.

Table 5-2: Summary of All Results

CAD System	Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7
NX	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	NA	Max: % 100.0 Avg: % 46.3	NA	NA	Max: % 0.0 Avg: % 0.0
Pro/E	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	NA	NA	NA	NA	NA
CATIA	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	NA	NA	NA	NA	NA
SolidWorks	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	Max: % 34.8 Avg: % 3.5	Max: % 100.0 Avg: % 46.3	Max: % 39.5 Avg: % 9.1	NA	Max: % 0.0 Avg: % 0.0
Inventor	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	Max: % 17.7 Avg: % 1.7	Max: % 3.3 Avg: % 0.0	NA	NA	NA
Hybrid Method	Max: % 0.0 Avg: % 0.0	Max: % 0.0 Avg: % 0.0	Max: % 2.3 Avg: % 0.2	Max: % 2.5 Avg: % 0.2	Max: % 2.2 Avg: % 0.1	Max: % 2.2 Avg: % 0.1	Max: % 2.3 Avg: % 0.1

5.6 **Test Case 7**

Test part 7 requires additional attention. This part was created to demonstrate the handling of global intersections. This test part produces a genus 1 surface and must be cut to become a genus 0 surface as was mentioned in chapter 4. Figures 46, 47, and 48 show the process of cutting the surface. However, It is extremely rare to find parts that exhibit this behavior and the global intersection can easily be avoided by modifying the part. Figure 49 shows the modified test part that was used to generate the offset surface.

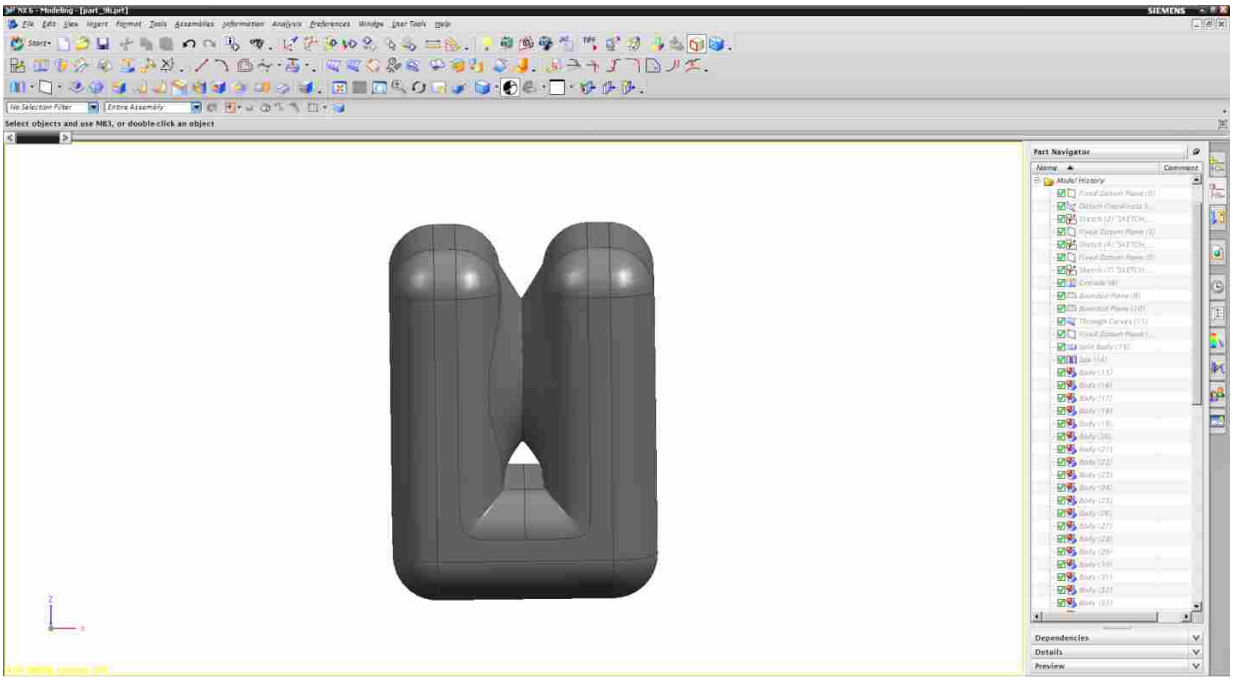


Figure 5-45: Hybrid Method Part 7

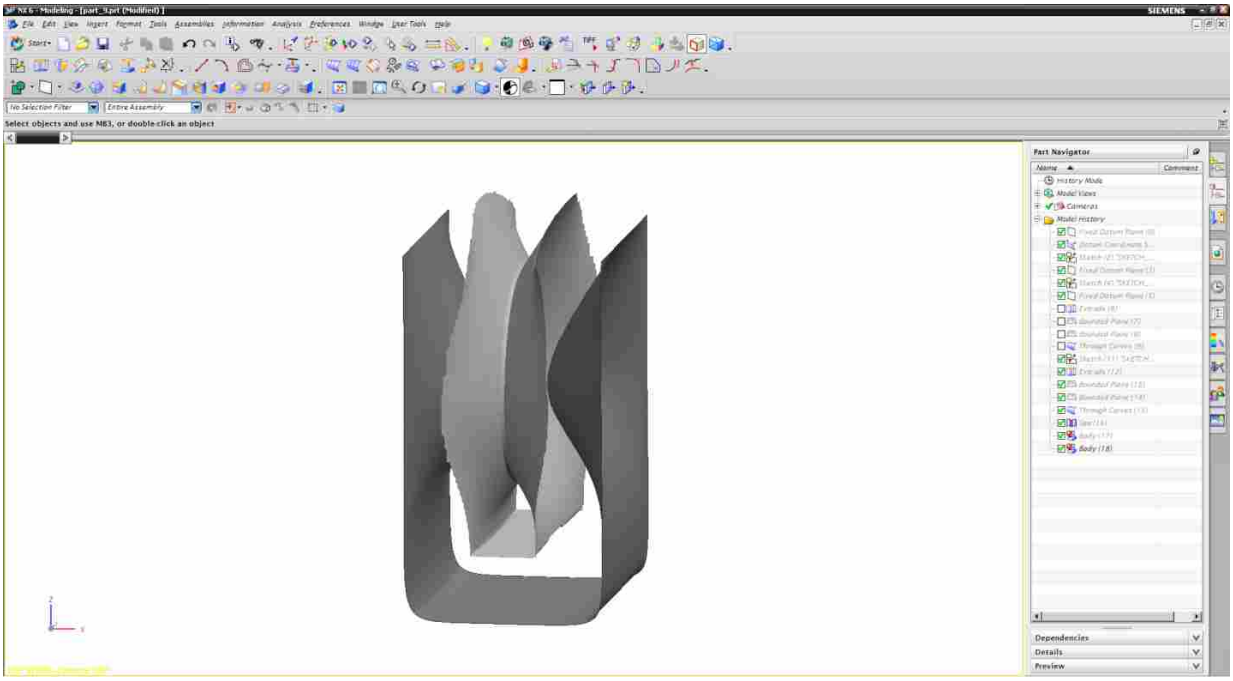


Figure 5-46: Genus 1 Surface

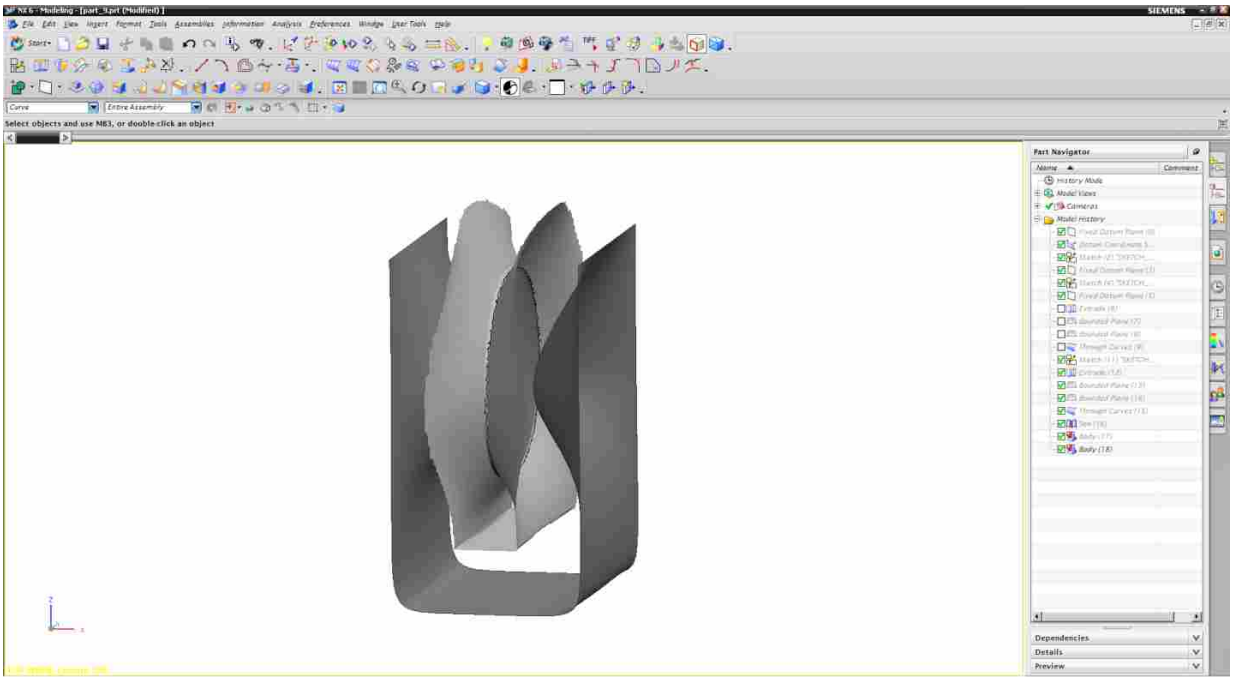


Figure 5-47: Cutting the Surface

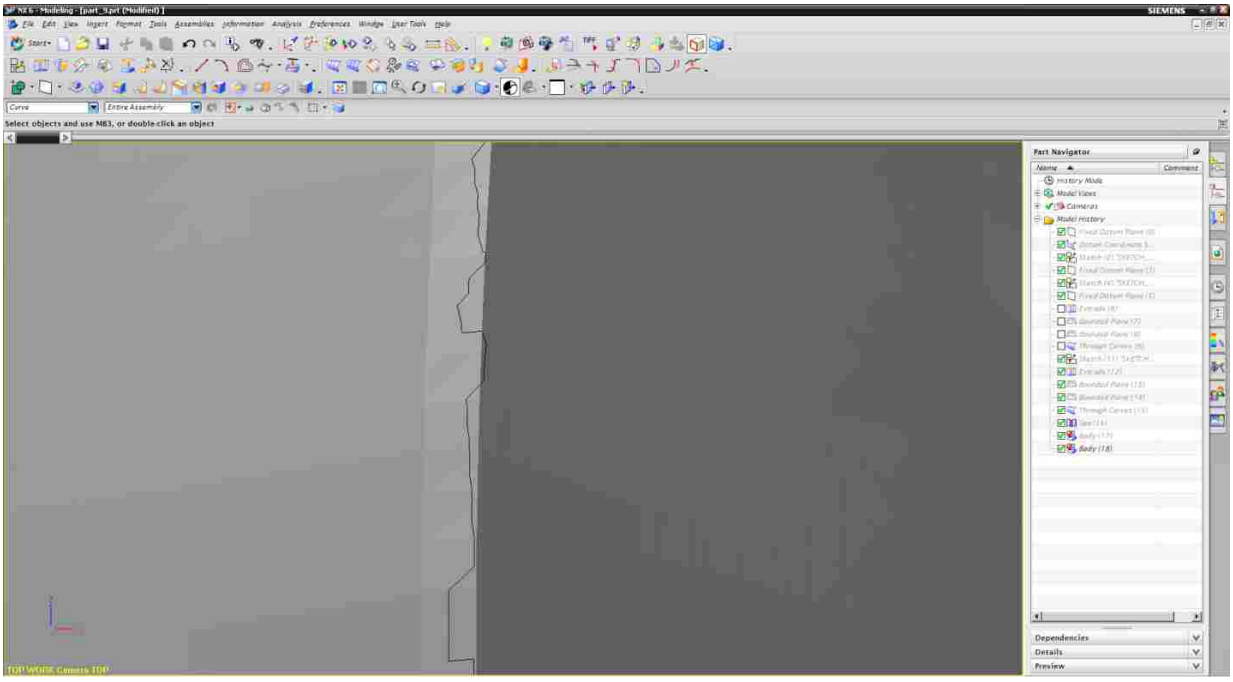


Figure 5-48: Cutting the Surface

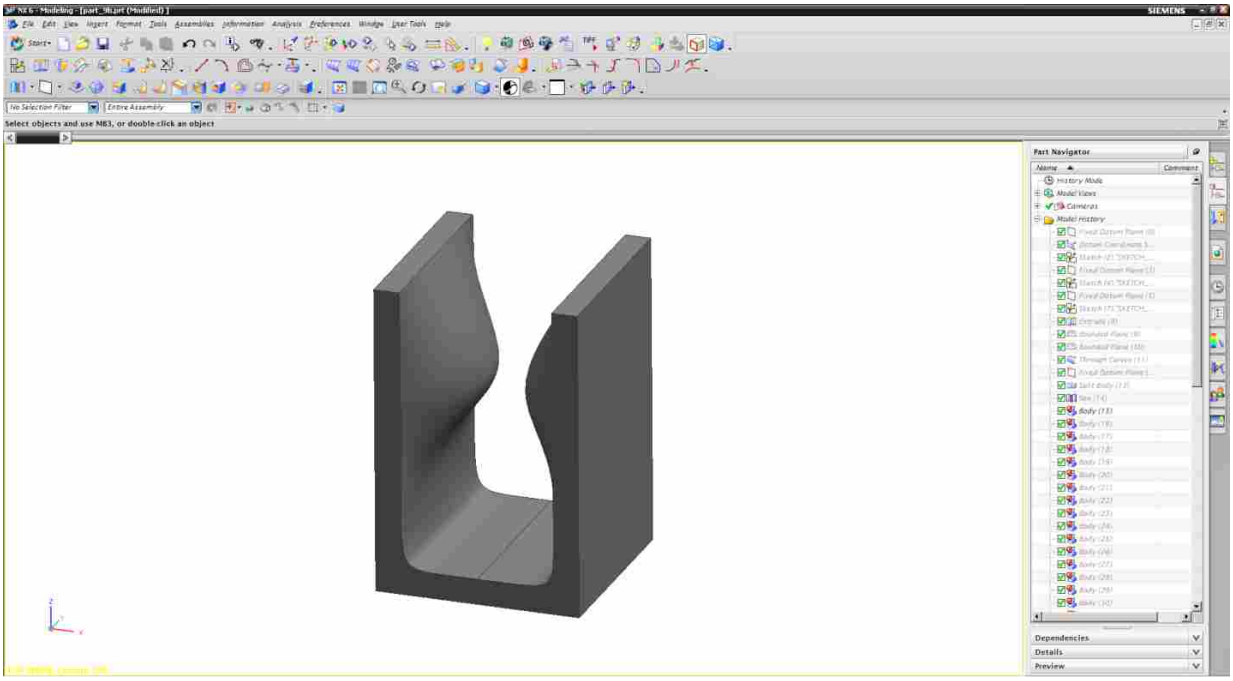


Figure 5-49: Test Part 7 Modified

5.7 Conclusion

The hybrid offset method has produced offset surfaces for complex models that commercial CAD systems could not handle. Based on these results, it can be certain that the hybrid method is of great value. There are of course drawbacks to this method. The speed at which the offsets were created was not an objective of this research. Therefore, the amount of time that it took to complete these offsets was not recorded. The total time to create an offset was on the order of magnitude of 30-45 minutes. The extremely long computation is a result of a high accuracy grid, computationally expensive calculations (e.g. minimum distance queries), and no optimization techniques. These results highlight the difference between the conventional parametric offset method and the computational method hybrid. The parametric method is faster, but the hybrid method is more robust.

6 CONCLUSIONS

As was discussed in the introductory chapter of this document, the goal of this research was to demonstrate a new method of creating offset surfaces. This method was to be validated by comparing the results of this new method with the results of leading CAD systems. As shown in the previous chapter, the hybrid method can offset surfaces that commercial CAD systems cannot. This method, however, is not perfect. There are many areas in which this method could be improved. The area of biggest improvement would be the speed. Applying optimizations that are found in the literature could greatly improve the performance of this new method. Multi-threading could also greatly improve the performance of this method. This process lends itself well to multi-threading, the Marching Tetrahedron portion of the method can be multi-threaded and the rebuilding of the individual faces could also be multi-threaded.

It is also important to note that this methods produces results that are different from CAD system offset tools. CAD systems produce offset surfaces, then extend and trim the surfaces against each other, where the hybrid method produces rounded corners. This is a drawback because many times the design requires a sharp corner and not a round. This difference however can also be an advantage. In certain applications, such as layered manufacturing (e.g. composites) or coating processes (e.g. investment casting shell) the rounded corners are more representative of reality. Future work could be focused on how to extend the hybrid method that

sharp corners could be persevered, thereby allowing the user to choose between corners and rounds.

Another area where the method could be improved is the type of surfaces the method can produce. NURBS surfaces represent the standard for free-form surfaces in CAD systems and were therefore the focus of this research. However, a new surface type known as T-Splines is gaining popularity in the CAD industry. This new surface type has a distinct advantage over NURBS surfaces. NURBS surfaces must be four sided, whereas T-Splines can take on any shape and any number of sides without having to be trimmed. By using T-Splines with this process, the generated offset faces could be represented by T-Splines and would not require any extrapolation of extra points.

These various improvements were outside the scope of this research, the goal was to prove the robustness of this method. Now that it has proven to be a valuable method, it can be improved upon.

REFERENCES

Braid, I., (1975) "The Synthesis of Solids Bounded by Many Faces" *Association of Computing Machinery*, 209-216

Cheernyaev, E., (1995) "Marching Cubes 33: Construction of Topologically Correct Isosurfaces" *Institute for High Energy Physics*

Coquillart, S., (1987) "Computing Offsets of B-Spline Curves" *Computer Aided Design*, 19(6), 305-309

De Boor, C., (1972) "On Calculating with B-Splines" *Journal of Approximation Theory*, 6, 50-62

Dziegielewski, A., Erbes, R., Schomer, E., (2010) "Real-Time Offset Surfaces" *In EuroCG Workshop On Computational Geometry*

Elber, G., Cohen, E., (1991) "Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces" *International Journal of Computational Geometry & Applications*, 1(1)

Elber, G., Grandine, T., Kim, M., (2009) "Surface Self-Intersection Computation Via Algebraic Decomposition" *Computer Aided Design*, 41, 1060-1066

Farin, G., (1983) "Algorithms for Rational Bezier Curves" *Computer Aided Design*, 15(2), 73-77

Farin, G., (1989) "Curvature Continuity and Offsets for Piecewise Conics" *ACM Transactions on Graphics*, 8(2), 89-99

Farin, G., (2002) *Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide*, London

Faux, I., Pratt, M., (1984) *Computational Geometry for Design and Manufacture*

Floater, M., (1992) "Evaluation and properties of the Derivatives of a NURBS Curve" *Mathematical Methods in CAGD and Image Processing*, 1-15

Floater, M., (1991) "Derivatives of Rational Bezier Curves" *Computer Aided Geometric Design*, 9(3), 161-174

Flutter, A., Todd, J., (2001) "A Machining Strategy for Toolmaking" *Computer Aided Design*, 33, 1009-1022

Forsyth, M., (1995) "Shelling and Offsetting Bodies*" Solid Modeling 95*, 373-381

Frisken, S., Perry, R., (2006) "Designing with Distance Fields" *ACM SIGGRAPH 2006*, 249-254

Jang, D., Park, H., Kim, K., (2005) "Surface Offsetting Using Distance Volumes" *International Journal of Advanced Manufacturing Technology*, 26, 102-108

Jones, M., Baerentzen, J., Sramek, M., (2006) "3D Distance Fields: A Survey of Techniques and Applications" *Visualization and Computer Graphics*, 12(4), 581-599

Ju, T., Losasso, F., Schaefer, S., Warren, J., (2002) "Dual Contouring of Hermite Data" *Association of Computing Machinery*, 339-346

Ju, T., Udeshi, T., (2006) "Intersection-Free Contouring on an Octree Grid" *Pacific Graphics*

Jung, W., Shin, H., Choi, B., (2004) "Self-Intersection Removal in Triangular Offsetting" *Computer Aided Design and Applications*, 1(1-4), 477-484

Kim, S., Lee, D., Yang, M., (2004) "Offset Triangular Mesh Using the Multiple Normal Vectors of a Vertex" *Computer Aided Design and Applications*, 1(1-4), 285-291

Kulczycka, M., Nachman, L., (2002) "Qualitative and Quantitative Comparisons of B-Spline Offset Surface Approximation Methods" *Computer Aided Design*, 34, 19-26

Kumar, G., Shastry, K., Prakash, B., "Computing Non-Self-Intersecting Offsets of NURBS Surfaces" *Computer Aided Design*, 34, 209-228

Levy, B., Petitjean, S., Ray, N., Maillot, J. (2002) "Least Squares Conformal Maps for Automatic Texture Atlas Generation" *ACM Transactions on Graphics (TOG),* 21(3), 362-371

Liu, S., Wang, C., (2009) "Duplex Fitting of Zero-Level and Offset Surfaces" *Computer Aided Design*, 41, 268-281

Liu, S., Wang, C., (2011) "Fast Intersection-Free Offset Surface Generation From Free-Form Models with Triangular Meshes" *IEEE Transactions on Automation Science and Engineering*, 8(2), 347-360

Lorensen, W., Cline, H., (1987) "Marching Cubes: A High Resolution 3D Surface Construction Algorithm" *Computer Graphics*, 21(4), 163-169

Maekawa, T., (1999) “An Overview of Offset Curves and Surfaces” Computer Aided Design, 31, 165-173

Malosio, M., Pedrocchi, N., Tosatti, L., (2009) “Algorithm to Offset and Smooth Tessellated Surfaces” *Computer Aided Design and Applications*, 6(3), 351-363

Overveld, K., Wyvill, B., (2004) “Shrinkwrap: An Efficient Adaptive Algorithm for Triangulating an Iso-surface” *The Visual Computer* 20, 362-379

Pavic, D., Kobbelt, L., (2008) “High-Resolution Volumetric Computation of Offset Surfaces with Feature Preservation” *EUROGRAPHICS*, 27(2)

Pekerman, D., Elber, G., Kim, M., (2008) “Self-Intersection detection and elimination in Freefrom Curves and Surfaces” *Computer Aided Design*, 40, 150-159

Piegl, L., Tiller, W., (1999) “Computing Offsets of NURBS Curves and Surfaces” *Computer Aided Design*, 31, 147-156

Piegl, L., Tiller, W., (1987) “Curve and Surface Constructions Using Rational B-Splines” *Computer Aided Design*, 19(9), 485-498

Piegl, L., Tiller, W., (1997) *The NURBS Book*, Germany

Pottman, H., (1995) “Rational Curves and Surfaces with Rational Offsets*” Computer Aided Geometric Design*, 12, 175-192

Rogers, D., (2001) *An Introduction to NURBS – With Historical Perspective*, London

Rossignac, J., Requicha, A., (1986) “Offsetting Operations in Solid Modelling” *Computer Aided Geometric Design*, 3, 129-148

Seong, J., Elber, G., Kim, M., (2006) “Trimming Local and Global Self-Intersections in Offset Curves/Surface Using Distance Maps” *Computer Aided Design*, 38, 183-193

Shah, J., Mantyla, M., (1995) *Parametric and feature-based CAD/CAM*, New York, NY

Shen, H., Fu, J., Chen, Z., Fan, Y., (2010) “Generation of Offset Surface for Tool Path in NC Machining Through Level Set Methods” *International Journal of Advanced Manufacturing Technology*, 46, 1043-1047

Thomassen, J., () “Self-Intersection Problems and Approximate Implicitization” *Computational Methods for Algebraic Spline Surfaces,* 155-170

Tiller, W., Hanson, E., (1984) “Offsets of Two-Dimensional Profiles*” IEEE CG&A*, 36-46

Treece, G., Prager, R., Gee, A., (1999) "Regularised Marching Tetrahera: Improved Iso-Surface Extraction" *Computers And Graphics*, 23, 583-598

Varadhan, G., Manocha, D., (2006) "Accurate Minkowski Sum Approsimation of Polyhedral Models" *Graphical Models*, 68(4), 343-355

Varadhan, G., Krishnan, S., Kim, Y., Manocha, D., (2003) "Feature-Sensitive Subdivision and Isosurface Reconstruction" *Proceedings of the 14th IEEE Visualization Conference*, 99-106

Wang, C., Manocha, D., (2013) "GPU-Based Offset Surface Computation Using Point Samples" *Computer-Aided Design*, 45, 321-330

Yin, K., Liu, Y., Wu, E., (2011) "Fast Computing Adaptively Sampled Distance Field on GPU" *Pacific Graphics* 2011

Yoo, D., (2009) "General 3D Offsetting of a Triangular Net Using an Implicit Function and the Distance Fields*" International Journal of Precision Engineering and Manufacturing*, 10(4), 131-142

Zhang, Y., Yu, M., (2011) "Computing Offsets of Point Clouds using Direct Point Offsets For Tool-Path Genreation" *IMechE*, 226(B), 52-65