2003

CAD model robustness assessment and repair

Armand Daryoush Assadi
Iowa State University

CAD model robustness assessment and repair

by

Armand Daryoush Assadi

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:
James H. Oliver, Major Professor
James E. Bernard
Judy M. Vance
Thomas J. Rudolphi
Bion L. Pierson

Iowa State University

Ames, Iowa

2003

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Armand Daryoush Assadi

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Dedication

To my parents, more excited than I about finishing, and to all my grandparents, whom I wish could be here to celebrate this accomplishment with me.

Table of Contents

Table of Contents iv

List of Figures vii

List of Tables x

List of Symbols xi

Acknowledgements xiii

Abstract xiv

Chapter 1. Introduction 1

1.1. Problem Statement 1

1.2. Motivation 1

1.3. Contribution 3

Chapter 2. Literature Review 4

2.1. History of CAD and Geometric Modeling 4

2.2. Data Exchange Standards 8

2.2.1. Historical Development of Data Exchange Standards 8

2.2.2. IGES 11

2.2.3. STEP 12

2.2.4. How STEP Addresses the Data Exchange Problem 14

2.3. Data Transfer Problems 15

2.3.1. Errors within the Original Model 16

2.3.2. Errors in the Transfer of the Model 16

2.3.3. Errors in Putting the Model Back Together 19

Chapter 3. Loop Closure Algorithm 20

3.1. Algorithm Components 23

3.1.1. Mapping 23

3.1.2. Inverse Mapping 23

3.1.3. Normalize 24

3.1.4. Form Trim Loops 24

3.1.5. Nearest Neighbor 25

3.1.6. Create Curve 28

3.1.7. Delete Curve 28

3.1.8. Split Curve 28

3.1.9. Truncate 29

3.1.10. Extend 30

3.1.11. Insert 31

3.1.12. Orient 33

3.2. Loop Closure Methodology 34

3.2.1. Pre-Processor 35

3.2.2. Processor 35

3.2.2.1. No Trim Curves 36

3.2.2.2. One Trim Curve 37

3.2.2.3. Two Trim Curves 38

3.2.2.4. More Than Two Trim Curves 39

3.2.3. Post-Processor 39

Chapter 4. Examples 41

4.1. Pole Errors 41
4.2. Gap Errors 45
4.3. Seam Errors 47
4.4. Degenerate Edge Errors 50
Chapter 5. Conclusions 53
Appendix A. NURBS 59
A.1. NURBS Curve 59
A.2. NURBS Surface 60
A.3. NURBS Curve Derivatives 61
A.4. Trim Curves 62
A.5. Mapping Parameter Space to Model Space Curves 64
A.6. Inverse Mapping Model Space to Parameter Space Curves 65
A.7. NURBS Curve Splitting 66
A.8. NURBS Curve Normalization 67
Appendix B. Types of Geometric Models 68
B.1. CSG 70
B.2. B-Reps 72
Appendix C. Implicit vs. Parametric Equations 75
Appendix D. Parametric vs. Variational Modeling 77
Appendix E. Topological Challenges in Surface Modeling 80
Appendix F. Sample IGES File 83
Appendix G. Sample STEP File 85
Bibliography 88

List of Figures

Figure 2.1 Direct data transfer situation. 9

Figure 2.2 Neutral file transfer situation. 9

Figure 2.3 Loss of accuracy. 17

Figure 2.4 Loss of representation structure. 18

Figure 3.1 Joining of curves with too small a tolerance value. 26

Figure 3.2 Joining of curves with too large a tolerance value 26

Figure 3.3 Grouping endpoints by nearest neighbor. 27

Figure 3.4 Truncating trim curves. 30

Figure 3.5 Extending trim curves. 31

Figure 3.6 Trim curve loops and the resulting hierarchy. 32

Figure 3.7 (a) C_1 contains C_2, (b) C_2 contains C_1, and (c) C_1 and C_2 are disjoint. 32

Figure 3.8 Relationship between two trim curve loops 33

Figure 3.9 The loop closure structure 34

Figure 3.10 (a) No trim loops and (b) complete boundary trim loop. 37

Figure 3.11 Spatial positions of one trim curve in parameter space. 37

Figure 3.12 Linear and non-linear interior and semi-interior trim curves. 38

Figure 3.13 Completed boundary trim loop. 38

Figure 3.14 (a) Proper vs. (b) improper orientation of trim curves within a trim loop. 39

Figure 4.1 Primitives with pole errors 42

Figure 4.2 Parameter space trim curves exhibiting a pole error 43

Figure 4.3 Model space trim curves exhibiting a pole error. 43

Figure 4.4 Parameter space trim curves without a pole error. 44

Figure 4.5 Model space trim curves without a pole error. 44

Figure 4.6 Primitives without pole errors after loop closure. 44

Figure 4.7 Motor housing unit with gap errors. 45

Figure 4.8 Parameter space trim curves exhibiting a gap error. 46

Figure 4.9 Model space trim curves exhibiting a gap error. 46

Figure 4.10 Parameter space trim curves without a gap error. 46

Figure 4.11 Model space trim curves without a gap error. 47

Figure 4.12 Motor housing unit without gap errors after loop closure. 47

Figure 4.13 Sheet metal part with seam errors. 48

Figure 4.14 Parameter space trim curves exhibiting a seam error. 48

Figure 4.15 Model space trim curves exhibiting a seam error. 49

Figure 4.16 Parameter space trim curves without a seam error. 49

Figure 4.17 Model space trim curves without a seam error. 49

Figure 4.18 Sheet metal part without seam errors after loop closure. 50

Figure 4.19 Converter surface with degenerate edge errors. 50

Figure 4.20 Parameter space trim curves exhibiting a degenerate edge error. 51

Figure 4.21 Model space trim curves exhibiting a degenerate edge error. 51

Figure 4.22 Parameter space trim curves without a degenerate edge error. 52

Figure 4.23 Model space trim curves without a degenerate edge error. 52

Figure 4.24 Converter surface without degenerate edge errors after loop closure. 52

Figure 5.1 Parameter space curves for unsolved faces. 57

Figure 5.2 Model space curves for unsolved faces. 57

Figure A.1 A NURBS curve. 60

Figure A.2 A NURBS surface. 61

Figure A.3 First derivative of a cubic NURBS curve computed at $u = 0.5$. 62

Figure A.4 Parameter space and model space trim curves. 63

Figure A.5 Mapping parameter space to model space. 64

Figure A.6 Inverse mapping from model space to parameter space. 66

Figure A.7 Splitting a NURBS curve. 66

Figure B.1 Wireframe, surface, and solid models of a bored cube block. 69

Figure B.2 A simple CSG model. 71

Figure B.3 Rectilinear block. 73

Figure B.4 Winged-edge. 73

Figure C.1 Unit circle centered at the origin. 75

Figure D.1 Parametric and variational design process. 77

Figure D.2 Types of constraints. 78

Figure D.3 Parametric vs. variational matrix approaches. 79

Figure E.1 Topological and parametric space of a plane. 81

Figure E.2 Topological and parametric space of a sphere (pole). 81

Figure E.3 Topological and parametric space of a cone (degenerate edge). 81

Figure E.4 Topological and parametric space of a cylinder (seam). 81

List of Tables

Table 5.1 Test files. 54

Table 5.2 Test file statistics. 55

Table B.1 Advantages and disadvantages of wireframe models. 69

Table B.2 Advantages and disadvantages of surface models. 69

Table B.3 Advantages and disadvantages of solid models. 70

Table B.4 Winged-edge data structure. 74

Table D.1 Advantages and disadvantages of the parametric approach. 79

Table D.2 Advantages and disadvantages of the variational approach. 79

List of Symbols

2D	Two-Dimensional
3D	Three-Dimensional
AECMA	Association Européenne des Constructeurs de Matériels Aeronautique
ANSI	American National Standards Institute
AP	Application Protocol
ASCII	American Standard Code for Information Interchange
B-Rep	Boundary Representation
BRT	Business Round Table
CAD	Computer Aided Design
CAD*I	Computer-Aided Design Interfaces
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAM-I	Computer-Aided Manufacturing – International
CSG	Constructive Solid Geometry
EDIF	Electronic Design Interchange Format
EIA	Electronic Industries Alliance
ESP	Experimental Solids Proposal
GE	General Electric
ICAM	Integrated Computer-Aided Manufacturing

IEC	International Electrotechnical Commission
IGES	Initial Graphics Exchange Specification
IPC	Institute for Interconnecting and Packaging Electronic Circuits
IPO	IGES/PDES Organization
ISO	International Organization for Standardization
ITI	International TechneGroup Incorporated
McAuto	McDonnell Douglas Automation
NBS	National Bureau of Standards
NEDO	National Economic Development Office
NURBS	Non-Uniform Rational B-Splines
PDDI	Product Data Definition Interface
PDES	Product Data Exchange Specification
SC4	Subcommittee 4
SET	Standard d'Echange et de Transfert
STEP	Standard for the Exchange of Product model data
TC184	Technical Committee 184
VDAFS	Verband der Automobilindustrie – Flachen-Schnittstelle
VDAIS	Verband der Automobilindustrie – IGES Subset
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VNS	Verfahrens Neutralen – Schnittstelle
XBF	Experimental Boundary File

Acknowledgements

Many people and organizations have helped, contributed, supported and critiqued this work. First of all, my parents have been there through the whole thing. Without their encouragement, prodding, funding, and antagonism, I may have never gotten to this point. I thank all my colleagues and instructors at Iowa State University, especially those from old ICEMT and new VRAC. I also send a special thanks to my major professor, Dr. James H. Oliver, for his patience.

Abstract

The transfer of CAD data among different CAD systems or subsequent downstream analysis applications is critically important to the acceleration of the product development cycle. Since each vendor has its own proprietary native file format, this transfer of data among differing systems is difficult at best. A proposed solution to the transfer problem was the development of agreed upon standard neutral file formats, such as IGES or STEP. However, each vendor writes its own "flavor" of IGES and STEP files that other applications may not understand.

The research presented in this dissertation bridges a gap between these systems by taking a native IGES or STEP file and either repairing the data or adding missing information so that a downstream application can properly interpret the model. This ensures that the receiving system gets a full and accurate NURBS-based representation: the original surfaces, the corresponding full complement of model space trim curves, and the corresponding full complement of parameter space trim curves. With all the information present, the receiving system can select the needed pieces to reconstruct the model.

Chapter 1. Introduction

1.1. Problem Statement

In today's Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), and Computer-Aided Engineering (CAE) world, the transfer of geometric data among CAD systems and subsequent downstream engineering applications, analysis tools, and visualization software is critically important to the acceleration of the development cycle. However, the success of such CAD systems and downstream applications is predicated on the receipt of accurate and complete geometrical models [1] [2]. Frequently, data files do not transfer between systems effectively [2]. This leads to increased costs in product development. The problem is to find a method that can assess the geometric data within these files and supply missing or modify inaccurate information so the data files can be usable by different systems [3].

1.2. Motivation

The challenge of how to exchange design and engineering data between systems has existed since CAD moved from academia to industry in the 1960s [4]. Even with the maturity of modern CAD systems, such as CATIA, Pro/ENGINEER, I-DEAS, Solid Edge and Unigraphics, their compatibility with other systems falls short of what is required by designers [2]. These interoperability issues arise because different systems define geometry differently and the vendors are not willing, for various marketing and intellectual property reasons, to make their systems interface directly with that of a competitor [5].

The need for geometric CAD data transfer is two fold: first, it is needed for collaboration and outsourcing among different companies; and second, it is needed for analysis among different software packages, created by various companies, within one company. Interoperability limits the amount of information that can be shared, creating excess cost in design and production. In 1999, it was estimated that interoperability within the automotive industry alone was costing $1 billion per year [6]. Pressure from large industries has led to some easing by vendors of their file formats, but not to the extent that is needed for the best possible cost reduction. Industries wasting tremendous amounts of resources need a much more efficient and reliable method for transferring their data.

The common solution that arose for the data transfer problem was the creation of neutral standard file formats, such as the Initial Graphics Exchange Specification (IGES) and, more recently, the Standard for the Exchange of Product model data (STEP), which each vendor would support. However, since the organizations established to oversee the development of the standards had little overlap with those actually involved in the development and implementations of the theory and algorithms, the standards themselves became open to individual interpretation that led to ambiguities and inconsistencies. Also, time-to-market pressures led many vendors that contributed to the development of the standards to lobby for broad definitions so they could meet the general intent of the specification with minimal effort and investment. As a result, translators created by different vendors vary greatly in terms of how they implement the standards. Thus, the files still do not transfer as effectively as would be desired [2].

Incomplete or erroneous CAD file translations drastically slow down the product development cycle. With only about 65 to 70 percent of data sets converting with no errors,

downstream users of these CAD models waste between 20 to 70 percent of their time reworking or totally recreating bad CAD data [3] [5]. These geometric model transfer problems blunt the competitive edge of enterprises by forcing them to compromise in either their choice of solutions or efficiency of the translation. Since CAD vendors are still not willing to share their file structures with each other, the best scenario is to find a way to make the neutral file formats work [5].

1.3. Contribution

This work introduces a methodology towards making the neutral file formats IGES and STEP work better for the current geometric data transfer problems. Analyzing the available geometric information within these files and repairing or adding missing information allows for a much improved rate of successful transfers between different systems, thus ensuring the reduction of the development cycle that such interoperability standards had originally envisioned.

Chapter 2. Literature Review

The exchange of geometric data among different CAD systems, engineering applications, analysis tools, and visualization software continues to be a challenge for achieving interoperability within industry. This chapter presents an overview of geometric modeling within CAD systems, the file formats that facilitate the exchange of such information, and the problems encountered in trying to do so.

2.1. History of CAD and Geometric Modeling

The beginning of geometric modeling can be traced back to the 1950s with the introduction of numerically controlled machine tools [7] [8]. Numerical part models created using piecewise linear or circular contour curves drove these machines. Although computationally fast, the complexity of these curves was limited. To overcome this limitation, conoids with parabolas were used, but at a computational cost [9].

By the 1960s, the aerospace, automotive, and ship building industries were suffering from these limitations [7] [9]. Given their design shapes, these industries had a strong interest in sculptured surface modeling. This stimulated research into the mathematics and applications of such surfaces. Bézier [10], Coons [11], de Casteljau [12], Ferguson [13] and Gordon [14] developed new curve and surface construction techniques based on parametric equations. Later, Cox [15], de Boor [16], and Riesenfeld [17] produced work based on B-splines [18]. All this work in parametric curve and surface modeling culminated in the Non-Uniform Rational B-Splines (NURBS) (Appendix A) as the *de facto* standard for curve and surface representation in CAD [19].

The parametric form (Appendix C) has the advantage of being able to easily compute and generate points and derivatives of a curve or surface [20]. Parametric equations also provide simple mathematical representations for complex shaped curves and surfaces, avoid problems resulting from coordinate system dependence (e.g., infinite slopes), and enable coordinate transformations of curves and surfaces to be performed simply [8].

In 1963, Ivan Sutherland developed the first interactive graphical system, called SKETCHPAD [21]. This work demonstrated the creation and modification of engineering drawings directly on a cathode ray tube. This led to the early use of CAD systems as drafting tools for creating engineering drawings, providing only two-dimensional (2D) functionality. This type of pure line and curve drawing of a model representation became known as "wireframe" (Appendix B) modeling.

In the 1970s, the desire to extend interactive 2D CAD systems to three dimensions emerged. However, this turned out to be more complicated than simply adding the third coordinate in the drawing representations. The three-dimensional (3D) wireframe models were often ambiguous, nonsense, or missing contour information. To address this situation, CAD systems began using parametric curve and surface representations so designers would have the flexibility of creating and visualizing complex models.

For designers interested primarily with surface (Appendix B) modeling, this was acceptable, but other applications required models that could represent not only surface features, but internal and external features as well. This led to the development of solid models (Appendix B). There were many different approaches to solid modeling, but due to their larger modeling domain, constructive solid geometry (CSG) [22] (Appendix B.1) and boundary representations (B-Reps) [23] (Appendix B.2) emerged as the most popular. With

the added sophistication of defining the spatial and topological relationships of collections of surfaces, and not just independent surfaces, the first solid modelers used implicit equations for defining CSG primitive objects and B-Rep faces.

Implicit equations (Appendix C) allow for simple classification of a point relative to a surface. Since surfaces are used to define CSG primitives and B-Rep faces, this can be extended to determine if a point lies inside, on, or outside a solid model. This was critical for determining operations such as union and intersection of two solid parts and also for analysis.

Industry also began to address the data transfer problems between different CAD systems through the development of IGES. At the time that it was developed, IGES was designed to handle data transfer for the current state of CAD systems, which were primarily implicit and parametric curves and surfaces.

During the 1980s, as computer graphics were advancing, CSG and B-Reps blended into hybrid CAD systems to take advantage of both solid model representation types. CSG representations were simple to construct using Boolean operation on pre-defined primitives and were concise in their representation, but were cumbersome for graphical output and some analysis purposes. B-Reps were useful for graphical applications, but were difficult to construct. Therefore, the emphasis of the early 1980's was maintaining a dual representation: a CSG-based user interface with underlying equivalent B-Reps at each stage of the modeling process [24] [25].

However, the predefined CSG primitives limited the complexity of the solid models. Designers wanted the flexibility to define solids bounded by parametric sculptured surfaces. Through approximately the mid- to late-1980s, implicit faces were replaced with parametric surfaces (most generally NURBS) and implicit edge loops by groups of parametric curves,

defined independently in 3D in the "model" space of the surface and/or within the 2D parameter space of the parent surface. By the end of the decade, the standard internal model representation for most CAD systems was hierarchical collections of B-Rep solids bounded by trimmed NURBS surfaces (Appendix A.4), even if this was not apparent through the user interface.

As solid modelers began to dominate the CAD market which surface modelers had done before, there was a need to transfer such data. The international development of STEP addressed the transfer of solid models while IGES continued to evolve to handle trimmed NURBS curves and surfaces.

The 1990s to the present witnessed the expanding capability of CAD. The Boolean operator-based user interfaces were replaced by methods that more closely reflected the needs of mechanical designers. For example, feature-based modeling such as fillets and rounds, became important. Even more profound was the advent of parametric and variational modeling (Appendix D) which allowed for the rapid modification of solid models. There was continued development of the user interface to make CAD systems more intuitive and less imposing for the designer. Despite all of this progress, the underlying representation of the model as a B-Rep bounded by trimmed NURBS surfaces has remained largely unchanged. This decade also saw the continued evolution of the data transfer standards that had begun in the 1980s for combating interoperability issues between the various CAD systems that were available.

2.2. Data Exchange Standards

Since CAD moved from academia into the mainstream in the 1960s, the problem of data exchange between different CAD/CAM/CAE systems has persisted [4]. This was due to the fact that vendors were not generally willing to make their file specifications available for various marketing and intellectual property reasons.

Three potential solutions exist to this problem; enforce the use of the same system throughout a project, build translators between each individual system, or develop a neutral data transfer specification to which each system adheres [26]. The first solution is impractical in today's international partnership environment since each organization has typically invested tremendous resources to their existing system. The second solution, at first, seems to show promise. However the number of translators grows exponentially by $n(n-1)$ where n is the number of systems, as shown in Figure 2.1. An upgrade to any one system means that $2(n-1)$ translators need to be upgraded and tested. The time and financial drain to provide such translators proves too costly. The third solution provides the most flexible and stable solution. This solution requires that only two new translators be developed for each system, for a total of $2n$ translators where n is the number of systems, as shown in Figure 2.2. A change in any one system requires only two translators to be upgraded and tested.

2.2.1. Historical Development of Data Exchange Standards

In the early 1970s, the American National Standards Institute (ANSI) Y14 Committee recognized the need for transferring data between different CAD systems. From this, a small voluntary group produced some drafts for a neutral transfer file format [27].

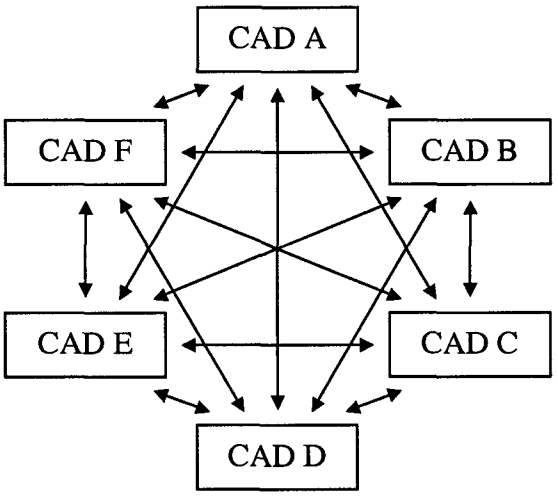


Figure 2.1 Direct data transfer situation [28].

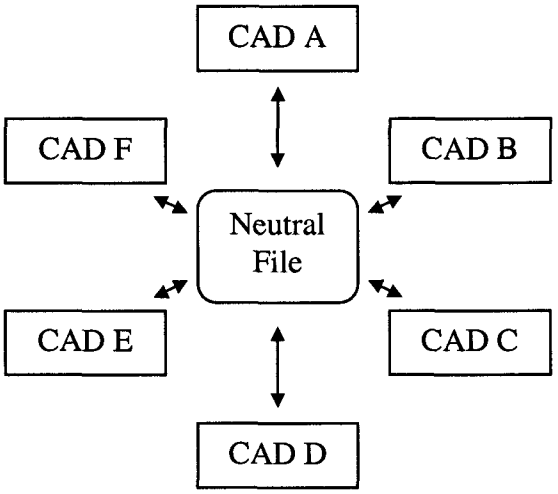


Figure 2.2 Neutral file transfer situation [28].

In the late 1970s, the Geometric Modeling Project of Computer-Aided Manufacturing – International (CAM-I) commissioned McDonnell Douglas Automation (McAuto) to prepare a file specification for solid modeling data using the ANSI Y14 Committee findings. CAM-I also commissioned Shape Data Ltd. to develop the procedural interface for solid modeling systems known as the Application Interface Specification [27].

At about the same time as the ANSI effort, the Air Force Integrated Computer-Aided Manufacturing (ICAM) Project contacted the National Bureau of Standards (NBS) (now the National Institute of Standards and Technology) to address the data exchange issue. The NBS developed the IGES file specification based on experience gained from Boeing's CAD/CAM Integrated Information Network and General Electric's (GE) Neutral Database. In the spring of 1980, the ANSI Y14 Committee adopted the IGES and McAuto specifications as part of their product data exchange standard. In January 1981, this standard was published and became the foundation for all future IGES specifications [27].

As CAD/CAM/CAE systems continued to evolve, the initial IGES specification failed to meet all the requirements that the industry needed. As a result, the IGES specification evolved to handle the new requirements, but new specifications based on IGES were also developed. However, the requirements for the industry have since surpassed the limits of IGES and the need for a new specification was apparent. The responsibility for the development of this new standard was given to the International Organization for Standardization (ISO), Technical Committee 184 (TC184: Industrial Automation Systems and Integration), Subcommittee 4 (SC4: Industrial Data and Global Manufacturing Programming Languages). They began the work in 1984 that led to the STEP standard to handle not only geometric data, but also all product data management information. Based on the limitations of the previously existing standards and their impending demise, most of the organizations behind the other file formats have contributed to the development of STEP.

2.2.2. IGES

By 1979, even though CAD systems were relatively new and few in number, frustrations were building with the inability to share data among the different systems. In September, this frustration was evident at the Air Force ICAM Industry Days meeting. A series of events and challenges between CAD vendors ComputerVision, Applicon, and Gerber, CAD users Boeing and GE, and a representative from the NBS led to the development of the first standard for CAD data exchange.

IGES (Appendix F) was the name for this standard, and it became the most widely accepted standard specification for geometric data transfer between different systems. Originally developed for the exchange of drafting information, such as those found in engineering drawings or blueprints, IGES evolved to include such specifications as 2D and 3D wireframe models, 2D and 3D curves and surfaces, CSG models, B-Rep models, finite element models, piping and electronic schematics, architectural drawings, engineering and construction elements and enhanced drafting entities for technical drawings.

Having started out under the direction of the IGES Organization and the NBS, IGES currently falls under the control of the National Computer Graphics Association, which administers the National IGES Users Group, and is part of the U.S. Product Data Association and the IGES/PDES Organization (IPO).

For all its advantages, IGES has three major shortfalls. First, the IGES specification is notorious for the ambiguity and imprecision of its definition, leaving implementers to make individual judgments as to the meaning of particular rules. Due to a lack of modeling standards, application functionality differences, invalid IGES data representations, and different interpretations of the standard, an entity produced by one vendor may not

necessarily be read by another that supports the same entity. However, a similar problem occurs when vendors only support particular entities, making it possible for two systems to have an entity mismatch that is not the fault of IGES but of the implementers. The IPO has published a provision of recommended practices to alleviate this problem, but it is still up to the implementers to follow them. Second, IGES files are very large because entities are defined both as a Directory Entry and Parameter Data; thus, the subsequent processing time is also large. This problem was addressed with the addition of binary and compressed ASCII options. Third, the IGES specification was not able to anticipate the manner in which CAD/CAM/CAE systems evolved and thus was not able to handle the changes such as the transfer of non-geometric data like product lifecycle information.

2.2.3. STEP

With the number of neutral data exchange formats increasing and the ability of CAD/CAM/CAE technology evolving to accomplish more complicated tasks, it was evident that the attempt to alleviate the problems of data transfer between systems was becoming more complicated. There was a need for a new standard that would go beyond the capabilities of the existing exchange formats. In July of 1984, the ISO TC184 SC4 began the development on a new standard designated as ISO 10303.

This standard was intended to provide a means of describing product data throughout its lifecycle far beyond what any other data transfer standard could do. It not only would define geometric shape, but also such data as assembly information, attributes, colors, configurations, decorations, features, materials used, relationships, tolerances, topology and weight in order to completely define a product for purposes such as design, engineering,

analysis, manufacturing, inspection, quality control testing, marketing, and product support. The design objectives of this standard needed to feature:

- Flexibility to permit expansion without invalidating existing portions of the standard.
- Efficiency for processing, communication and storage.
- Rigorous and formal documentation.
- The minimum possible set of data elements.
- Separation of data content from physical format.
- A logical classification of data elements.
- Compatibility with other existing relevant standards.

In March of 1985, this new standard was informally named STEP (Appendix G). In December of 1988, at a meeting in Tokyo, the basic structure and requirements of the standard were discussed and modified and subsequently distributed for international balloting. This initial version of STEP contained provisions for such things as curves and surfaces, B-Reps, CSG, tolerances, finite element modeling, drafting, machining, quality assurance, data management, and the EXPRESS language models. In June 1989, at a meeting held in Frankfurt, Germany, the results of the balloting were presented and the overall vote was to disapprove. After feedback was received, researchers and industry experts worked to define the standard better. In October 1991, ten parts were submitted for international balloting. When attendees gathered for a conference in Turin, Italy in 1993 and gave STEP their vote of confidence, it marked the long-awaited beginning of a new era in data exchange. In December 1994, the initial release of STEP consisted of twelve parts, including two Application Protocols (APs), approved as international standards. Today, STEP has twelve international standard APs and several more in the works.

STEP is a worldwide effort of 28 participating or observing nations such as Belgium, Canada, China, France, Germany, Italy, Japan, Poland, Sweden, Switzerland, the United

Kingdom and the United States. This international community is comprised of industry members, government agencies, and software vendors with most contributing to two major consortiums of STEP development, PDES, Inc. in the United States and ProSTEP in Europe. PDES, Inc. members include organizations such as Boeing, General Motors, IBM, PTC, and Raytheon. These organizations have developed STEP incorporating experiences gained from the development of other organizations and national standards such as IGES, PDDI [27], SET [29], VDAFS [30], PDES [31], and CAD*I [32] with additional contribution from VDAIS [33], AECMA [34], XBF [35], ESP [36], EDIF [37], VNS [38], VHDL [39], IEC TC3 [40], IPC-D-350 [41], EIA [42], NEDO [43] and BRT [44].

With an international consortium contributing to the development of STEP under the direction of the ISO, STEP is considered to be the key to interoperability among CAD/CAM/CAE systems. Most, if not all, other standards will become obsolete as STEP continues to mature and address the problems and limitations of these other standards.

2.2.4. How STEP Addresses the Data Exchange Problem

STEP is intended to be more than the next generation of IGES. The objective of STEP is to describe all data for a product throughout its lifecycle independent from any particular system. This is the major difference between STEP and most other standards that deal only with particular application areas.

STEP offers data integration capability that no other standard offers. Data integration ensures the single definition of information describing product design, manufacturing, and lifecycle support, thus eliminating redundancy and the problems associated with redundant information.

STEP uses a formal modeling language called EXPRESS to combat the ambiguous nature of the other standards. Making use of a formal language enables precision and consistency of representations and facilitates the growth of new implementations.

STEP uses a three-layer architecture enabling the definition of multiple applications and implementations. Implementation methods comprise the first layer and relate models to the EXPRESS language and then onto the physical file. Resource information comprises the second layer and provides context-independent information about such things as geometry, topology, and product structure. Application protocols comprise the third layer and contain information related to a particular application area such as automotive or sheet metal design. To ensure compliance, STEP provides a series of abstract test suites within a conformance testing methodology. This allows the STEP standard to continually grow, even in ways that are not foreseen.

STEP also provides other advantages such as providing complete support for solids such as CSG and B-Rep and a precise machine-readable specification. Finally, STEP is an international specification developed by users, not vendors. User-driven standards tend to be results-oriented where as vendor-driven standards tend to be technology-oriented. Therefore, STEP will theoretically survive changes in technology making it the solution for long term archiving of product data.

2.3. Data Transfer Problems

The success of any data exchange among different systems depends on a variety of factors. The model must initially be without defect within the originating CAD system, then the transfer process must work perfectly, and finally the receiving system must be able to

reconstruct the model from the information. However, experience has shown that this often is not the case.

2.3.1. Errors within the Original Model

The first problem area occurs within the originating CAD system in which the solid model may be ill defined. There are typically three types of errors that occur in the definition of solid models: structural errors, accuracy errors, and realism errors [45]. Structural errors violate the rules of solid modeling systems for what constitutes a valid model, such as free edges of a manifold solid or a misdirected normal vector. Accuracy errors occur when solid modeling systems approximate the curves resulting from surface intersections of a solid model. Defining such intersections explicitly is not always possible; therefore minimizing the standard deviation of such approximations is the best that can be hoped for. Realism errors constitute any artifact that has no physical meaning in a model such as transition cracks and sliver faces.

2.3.2. Errors in the Transfer of the Model

The second problem area occurs when transferring the model among CAD/CAM/CAE systems using a neutral file format. These errors are divided into three types: neutral file specifications, pre- and post-processors of CAD systems, and the general nature of CAD systems [46].

A neutral file specification that is ambiguous and incomplete leads to CAD vendors interpreting the specification differently. This causes problems with the types of entities that are supported and produced. This is the type of problem that led to the development of other specifications following IGES, such as VDAFS and SET.

Pre- and post-processors of CAD systems are often implemented imperfectly. Complex standards are difficult to implement leading to only a portion of the standard being supported. Also, programming errors can lead to incorrect implementation of the pre- and post processors. This type of problem led to the development of the abstract test suites and conformance testing methodology of STEP.

Most data transfer problems are attributed to the fundamental differences of CAD systems. These differences determine the amount of data that is lost during the exchange process. The differences of CAD systems can be classified as follows: accuracy, representation structure, mathematical description and geometrical representation [46] [47].

The different types of number precision used within a CAD system causes accuracy errors. For example, one system may use single precision while another system uses double precision. This type of error consistently represents the model in the receiving system, but with a general loss of accuracy, as shown in Figure 2.3.

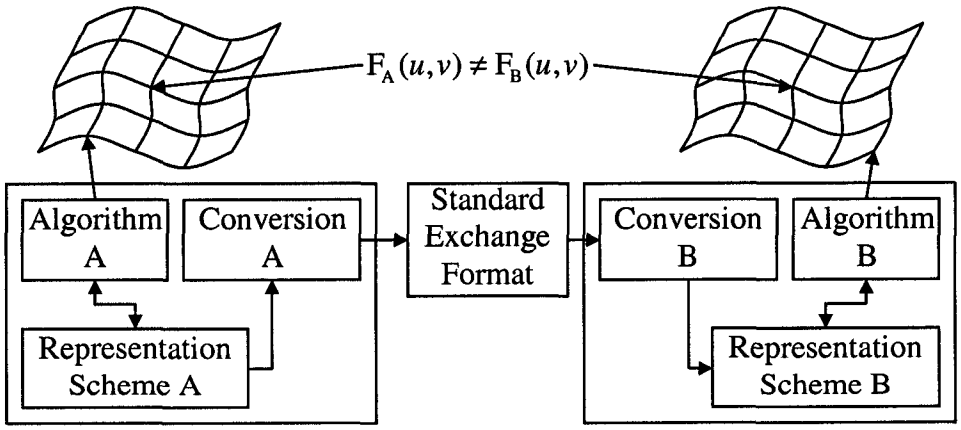


Figure 2.3 Loss of accuracy [47].

The representation structure of CAD systems may differ. For example, the surface grouping mechanisms of one system may differ from that of another systems resulting in a restructuring of model content. Again, this type of error consistently represents the model in the receiving system, but with a loss of model content, as shown in Figure 2.4.

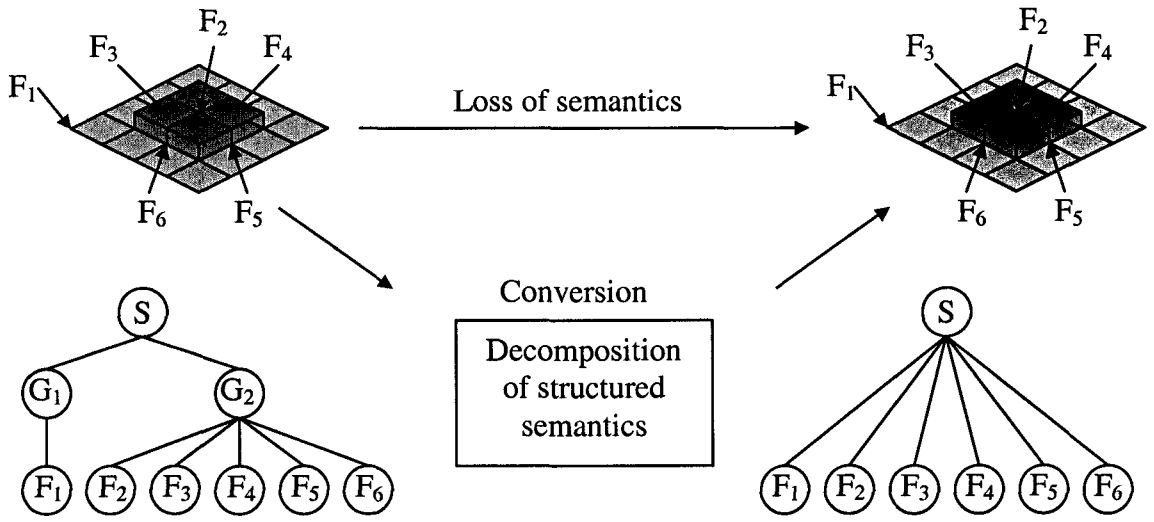


Figure 2.4 Loss of representation structure [47].

The mathematical description of CAD systems may also differ. For example, one system may use Bézier curve and surface representations for its models while another uses B-spline curve and surface representations for its models. This type of error will represent the model in the receiving system if the mathematical description of the originating system can be converted into the mathematical description of the receiving system. In most modern CAD systems this is no longer a problem with the acceptance of NURBS as the *de facto* standard for CAD modeling.

CAD systems use different kinds of geometrical representation for their models. Some systems might use CSG while others use B-Reps. Only if there is a conversion between the

different representation types will a model transfer properly. However, with the popularity of B-Reps to represent solid models, this type of geometrical representation problem between systems is lessening. Other types of geometry representation errors fall into two categories: individual surface representation errors and adjacent surface relationship errors [48].

Surface representation errors are usually caused by incorrect trim curves. Trim curve errors commonly occur with surfaces that contain poles, seams, or degenerate edges (Appendix E). For trim loops to be correct they must meet the following criteria:

- Trim loops must be continuous
- Trim loops must yield a bounded active surface
- Trim loops must lie completely within the parametric space
- Trim loops must not intersect themselves

Surface relationship errors may be self-intersecting surfaces, small gaps between surfaces, or overlaps between surfaces. Catastrophic self-intersecting surfaces are rarely seen. Small gaps or overlaps between surfaces may be corrected through correction of the surface's trim curves. If not, the surfaces need to be mended [49] [50] [51] [52] [53] [54].

2.3.3. Errors in Putting the Model Back Together

The third and final problem area occurs within the receiving CAD systems where the solid model again may be ill defined. The same type of problems may appear in the receiving system as they do in the originating or sending system: structural errors, accuracy errors, and realism errors. These types of errors are produced for the same reasons as in the originating system.

Chapter 3. Loop Closure Algorithm

By far the most common source of errors encountered in transferring contemporary surface and solid models are surface representation errors caused by incomplete and or misinterpreted trim loop. Despite their prevalence and importance, neither IGES nor STEP imposes a rigorous specification for trimmed surfaces. For example, trim curves (Appendix A.4) may be defined in the coordinate system of the parent surface (model space) or within the 2D parametric domain (parameter space) of the parent surface. Successive curves forming a trim loop need not be topologically ordered, and need not be closed. In some CAD systems, the boundaries of the parametric domain are implicitly assumed to be trim curves if not otherwise defined; while in others, loops must be explicitly defined. Some systems require trim curves to be defined only in model space while others require both model space and parameter space trim curves.

Thus, the core contribution of this work is a loop closure algorithm that assesses the quality of an IGES or STEP trimmed surface definition and repairs and augments the data so that a receiving system is more likely to be successful in properly interpreting it.

To assist with understanding the loop closure algorithm, the following terminology and assumptions are used:

- A trim curve is an individual parameter space or model space curve.
- A composite trim curve is a collection of trim curves linked together.
- A trim loop is a trim curve or composite trim curve that closes on itself.
- A parent surface is the base surface on which trim curves and loops are defined.

The primary assumption made by the loop closure algorithm is that the parent surface NURBS definition is complete and accurate. If there is no underlying surface to deal with,

the algorithm cannot work. Also, its parameter and model space curves are not assumed to be ordered or oriented. They are initially treated by the loop closure algorithm as a general collection of curves.

One final point of interest, although the loop closure algorithm accepts parameter and model space curves, all analysis is done within parameter space. This is done for the simplicity of working in a bounded 2D workspace rather than an unbounded 3D workspace. For this reason, if the input is limited to only model space curves, they must be inversely mapped to parameter space (Appendix A.6).

The motivation of this work, as stated earlier, is to get the standard data exchange formats IGES and STEP to transfer the needed geometric information to another system correctly. To that end there needs to be an assumption that the IGES and STEP files are valid according to their specification since this work is not a "how-to" for exporting valid geometric data into a valid file structure. What this work focuses on is the current standard for the representation of surface and solid models; trimmed NURBS surfaces. Specifically, it focuses on correcting trim curve errors caused by number precision accuracy or by the different geometric representation types of CAD systems.

To solve any trim curve errors, there needs to be a validity and consistency check of the geometric information. The validity check ensures that the trim curves are properly defined and that a closed trim loop is created. The consistency check ensures that a full trimmed NURBS surface definition is provided, meaning there exists a surface, a full complement of model space curves, and a full complement of parameter space curves.

Validity ensures that the trimmed NURBS definitions provided by the IGES or STEP file are valid and complete. The validity checks are that an underlying NURBS surface is

provided and that any provided models space and parameter space curves are valid. After checking each individual definition, the trim curves are checked to see if they form a valid trim loop. If not, closed trim loops need to be formed.

For consistency, there are four possible combinations of a trimmed NURBS surface definition that an authoring system may create: 1) definition of the surface only, 2) definitions for the surface and its corresponding model space curves, 3) definitions for the surface and its corresponding parameter space curves, or 4) definitions for the surface and its corresponding model space and parameter space curves. A target system will also require that the imported data be in one of these four combinations. However, if there is an information type mismatch, the target system will not be able to recreate the geometric information. For example, an authoring system provides surface and model space curve definitions, but a target system requires surface and parameter space curve definitions. Therefore, the consistency check, given a valid underlying NURBS surface, maps any of the missing model space or parameter space curves.

To provide for a robust representation of models, the loop closure algorithm is comprised of several specific functions to achieve this goal. These functions include creating, deleting, and splitting of curves; normalizing curve and surface definitions; truncating or extending curves to form an endpoint intersection; mapping or inverse mapping of parameter and model space curves; forming composite trim curves from trim curves; joining disjoint trim curves and composite trim curves; and orienting trim loops and their corresponding trim curves in the proper trimming direction.

The loop closure algorithm is described in three parts: the pre-processor, the processor, and the post-processor. The pre-processor takes care of any initial inverse mapping that is

needed and normalizes all NURBS definitions The processor is the heart of the loop closure algorithm where robust trimmed surface definitions are created. The post-processor orients the trim loops properly before the repaired surface is passed on.

3.1. Algorithm Components

This section describes the components that make up the loop closure algorithm. Readers not familiar with NURBS curve and surface modeling are encouraged to review Appendix A.

3.1.1. Mapping

The mapping function maps a parameter space curve to its model space equivalent. This function takes as input a parameter space curve $\mathbf{C}_p(t) = \{u(t),v(t)\}$ and its parent surface $\mathbf{S}(u,v)$ and returns the corresponding model space curve $\mathbf{C}_m(t) = \{u(t),v(t)\}$. For a further detailed discussion about mapping, refer to Appendix A.5. Pseudo code for this function is as follows:

Mapping $(\mathbf{C}_p(t), \mathbf{S}(u,v))$
  if $\mathbf{C}_p(t)$ = NULL or $\mathbf{S}(u,v)$ = NULL **then**
    return NULL
  for t = [0,1] increment along $\mathbf{C}_p(t)$ by Δt
    solve $\mathbf{S}_i(u(t_i),v(t_i))$
  return $\mathbf{C}_m(t)$ = interpolate($\mathbf{S}_i$)

3.1.2. Inverse Mapping

The inverse mapping function maps a model space curve to its parameter space equivalent. This function takes as input a model space curve $\mathbf{C}_m(t) = \{u(t),v(t)\}$ and its parent surface $\mathbf{S}(u,v)$ and returns the corresponding parameter space curve

$\mathbf{C}_p(t) = \{u(t), v(t)\}$. For a further detailed discussion about inverse mapping, refer to Appendix A.6. Pseudo code for this function is as follows:

Inverse Mapping ($\mathbf{C}_m(t)$, $\mathbf{S}(u,v)$)
  if $\mathbf{C}_m(t)$ = NULL or $\mathbf{S}(u,v)$ = NULL **then**
    return NULL
  for t = [0,1] increment along $\mathbf{C}_m(t)$ by Δt
    solve (u_i, v_i)
  return $\mathbf{C}_p(t)$ = interpolate((u_i, v_i))

3.1.3. Normalize

The normalize function normalizes the parametric range of curves and surfaces to [0,1] (Appendix A.8). This puts the curve and surface definitions into their preferred default state. With the proper form for the definitions, the data is ready to proceed to the main processor. This function takes as input the curve and surface definitions and normalizes the knot vector and the parameter space control points then returns the same curves and surfaces. Pseudo code for this function is as follows:

Normalize (curve/surface)
  compute parametric range of knot vector(s)
  use parametric range to reduce knot vector(s) range to [0,1]
  use parametric range to reduce parameter space control points range to [0,1]
  ***return** curve/surface*

3.1.4. Form Trim Loops

The loop closure automatically assembles successive trim curves together to form composite trim curves until a trim loop has been formed. As many composite trim curves as possible are initially constructed whether they form trim loops or not. From the collection of trim curves for each face, one is selected to analyze. There are three possibilities for this trim

curve: it forms a trim loop by self-enclosing; it can be joined to a composite trim curve (and possibly completing a trim loop); or it does neither, and a new composite trim curve must be started. For the trim curve to form a closed loop, the leading and trailing endpoints of the curve must have the same coordinate value. For a trim curve to be appended to a composite curve, either the leading or trailing endpoint of the trim curve must be the same coordinate value as the leading or trailing edge of the composite trim curve. When the trim curve does not form a trim loop or adjoin to a composite trim curve, it is a stand-alone curve at this point forming its own composite trim curve. After this trim curve has been analyzed, it may be removed from the inputted list of trim curves. Linked lists are used as the mechanism to hold the trim curve information. Pseudo code for this operation is as follows:

```
Form Trim Loops (curves)
if the number of curves = 0 then
    return NULL
while the number of curves ≠ 0 do
    select a trim curve C_T from the curves
    if C_T is a closed curve then
        create a new trim loop C_L
    else if C_T is appendable to a composite trim curve C_CT then
        append C_T to C_CT
    else
        create a new composite trim curve C_CT
    remove C_T from curves
for each composite trim curve C_CT1 do
    for each composite trim curve C_CT2 ≠ C_CT1 do
        if C_CT2 is appendable to C_CT1 then
            append C_CT2 to C_CT1
            remove C_CT2
return the composite trim curves and trim loops
```

3.1.5. Nearest Neighbor

To create trim loops, trim curves (and composite trim curves) must be linked together to form closed loops. The approach of connecting endpoints that are closer than a

predetermined constant tolerance ε does not always lead to accurate results. A tolerance that is too small may not link some trim curves while a tolerance that is too large may link the wrong trim curves. Figure 3.1 shows a situation where a tolerance that is too small will not join all the appropriate trim curves. When curves *a* and *b* are compared, the distance d_1 between their closest endpoints is greater than the tolerance value ε_1 and are thus appropriately not connected. When curves *b* and *c* are compared, the distance between their closest endpoints d_3 is less than the tolerance value and they are appropriately joined. However, when curves *a* and *c* are compared, the distance d_2 between their closest endpoints is greater than the tolerance value and they are inappropriately not joined. Figure 3.2 shows a situation when the tolerance value is too large and joins the wrong trim curves together. When curves *a* and *b* are compared, the distance d_4 between their closest endpoints is less than the tolerance value ε_2 and are incorrectly joined together omitting the position where curve *c* should be placed. Therefore, different tolerances are needed for different areas of parameter space.

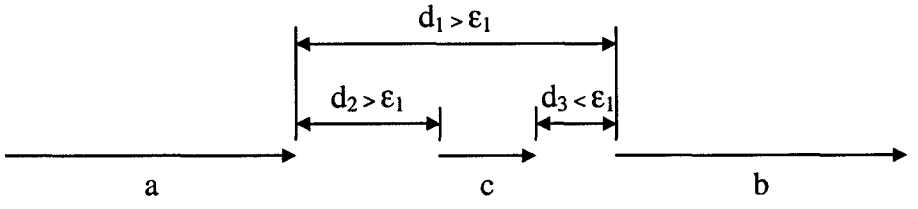


Figure 3.1 Joining of curves with too small a tolerance value.

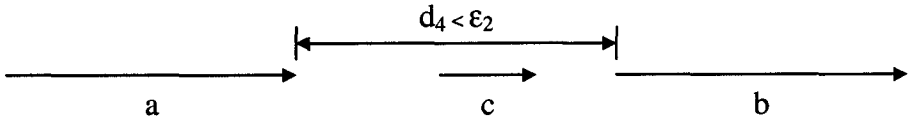
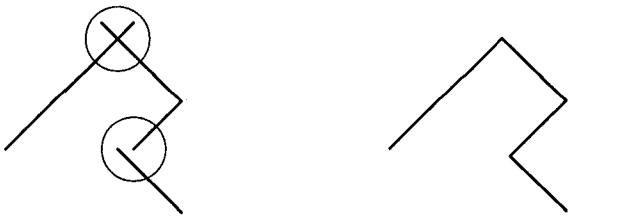


Figure 3.2 Joining of curves with too large a tolerance value.

The nearest neighbor approach is based on the entity linking and endpoints clustering method developed by Shpitalni and Lipson [55] [56]. This approach computes the minimum distance from every trim curve endpoint to every other trim curve endpoint, excluding itself. If two endpoints are the same shortest distance from each other, they are joined together. With this method, the endpoint wants an answer to the question, "You are my closest point. Am I your closest point?" If the answer is yes, it is likely that these two endpoints need to be joined. Formally, this is stated as:

for *each endpoint* P_i ***do***
    for *each endpoint* $P_j \neq P_i$ ***do***
        compute the distance d_{ij} *between* P_i *and* P_j
get shortest distance(s) d_{ijmin} *and corresponding endpoint(s)* P_{jmin}
for *each endpoint* $P_k \neq P_{jmin}$ ***do***
    compute the distance d_{jk} *between* P_{jmin} *and* P_k
    get shortest distance(s) d_{jkmin} *and corresponding endpoint(s)* P_{kmin}
if $d_{ijmin} = d_{jkmin}$, $d_{ijmin} <$ **tol**, *and* $P_i \neq P_{kmin}$ ***then***
    join P_i *and* P_{kmin}

This is repeated until no more trim curves are joined together. It is important to keep in mind that the nearest neighbor function is used only to solve for slight gaps between trim curves. It is not intended to solve the joining of two trim curves with a vast distance between them. Therefore, the distance is limited by some small upper bound. Figure 3.3 shows trim curves joined together derived using this method.



(a) Disjoint trim curves (b) Joined trim curves

Figure 3.3 Grouping endpoints by nearest neighbor [55].

3.1.6. Create Curve

The create curve function creates a linear parameter space NURBS curve (see Appendix A.1) between two given points. These points are the endpoints of the curve as well as its control points. Since a parameter space curve is being created, it must lie completely within the bounded region of parameter space. This is done with a simple check on the two endpoints. This function takes as input the two points and returns a linear curve. Pseudo code for this function is as follows:

```
Create Curve (point1, point2)
    if point1 or point2 is not bounded parametrically then
        return NULL
    create a curve between point1 and point2
    return the curve
```

3.1.7. Delete Curve

The delete curve function removes a given parameter space NURBS curve and provides all the necessary memory management and cleaning that is needed. This function takes as input the curve to be deleted. Pseudo code for this function is as follows:

```
Delete Curve (curve)
    if there is no curve then
        return NULL
    remove the curve
    clean up and memory management
```

3.1.8. Split Curve

The split curve function splits a NURBS curve (see Appendix A.7) by inserting knots at the desired parametric split location and creating the new control points so that the split curves coincide with the original un-split curve. This function takes as input the parameter

space curve definition that is to be split and the parametric value of where the split should occur and returns the two new NURBS curves. Pseudo code for this function is as follows:

Split Curve (curve, value)
if there is no curve or value ***then***
return *NULL*
split the given curve into two curves
return *the two curves*

3.1.9. Truncate

The truncate function finds either the (generally unlikely) self-intersection of one trim curve, or the intersection of two different trim curves, and deletes the overhanging portion of the curve. In general, given two trim curves $\mathbf{C}_1(s)$ and $\mathbf{C}_2(t)$, minimizing the function $f(s,t) = \mathbf{C}_1(s) - \mathbf{C}_2(t) = 0$ by Newton iteration solves the intersection point. Each curve exhibiting an intersection is then split at that intersection point using the split curve function. The curve that is significantly shorter by some predefined criteria than the other curve is deleted, as shown in Figure 3.4. This function takes as input either one or two trim curves, and returns either one or two truncated curves. Pseudo code for this function is as follows:

Truncate (curve(s))
if there is no curve(s) or point ***then***
return *NULL*
find the intersection point using Newton iteration
split the curve at the intersection point by calling **Split Curve (curve, point)**
delete the shorter curve by calling **Delete Curve (shorter curve)**
repeat for the other intersection
return *curve(s)*

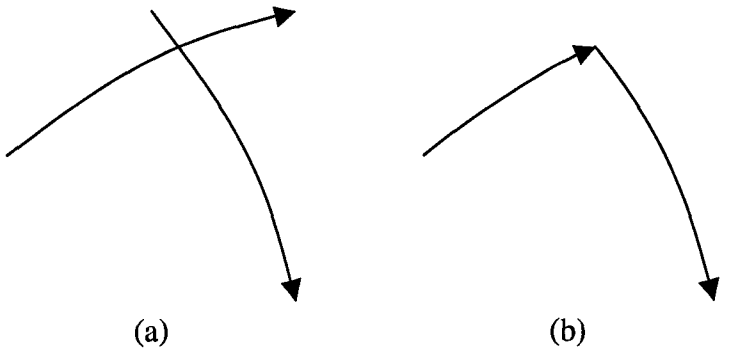


Figure 3.4 Truncating trim curves.

3.1.10. Extend

The extend function seeks an intersection point of parameter space curve(s) based on two endpoints and the tangents (Appendix A.3) of those endpoints. In general, given two trim curves $\mathbf{C}_1(s)$ and $\mathbf{C}_2(t)$, the intersection of $\mathbf{C}_1(1)+a\mathbf{C}_1'(1)$ and $\mathbf{C}_2(0)+b\mathbf{C}_2'(0)$, where $\mathbf{C}_1'(s)$ and $\mathbf{C}_2'(t)$ are the tangents to the curves, finds the point of extension.

If the intersection occurs within parameter space, new non-degenerate linear NURBS curves are created between the endpoints and the intersection point, as shown in Figure 3.5. New segments are created instead of modifying the existing curve(s) so that their original curve(s) definition is not altered in any way. If no intersection is found or if the intersection occurs outside of surface parameter space, the two endpoints are directly connected by a new linear NURBS segment. This function takes as input the two endpoints and their respective tangents and returns new curve(s) connecting those endpoints. Pseudo code for this function is as follows:

Extend (endpoint1, endpoint2, tangent1, tangent2)
extend line from **endpoint1** *along* **tangent1**
extend line from **endpoint2** *along* **tangent2**
compute the **intersection point** *of the two lines*
if *intersection exists and is within parameter space* ***then***
create a new curve by calling **Create Curve (endpoint1, intersection point)**
create a new curve by calling **Create Curve (endpoint2, intersection point)**
else
create a new curve by calling **Create Curve (endpoint1, endpoint2)**
return *curve(s)*

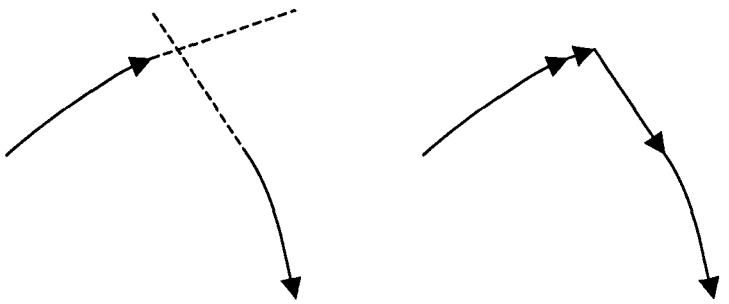


Figure 3.5 Extending trim curves.

3.1.11. Insert

The insert function, borrowed from [57], builds a containment-based hierarchy of trim loops, as shown in Figure 3.6. Each node in the hierarchy is a list of trim loops and each trim loop may refer to another list of trim loops that are contained with it. Since the trim curve loops are not allowed to intersect there are only three possible relationships between any two trim curve loops $\mathbf{C_1}$ and $\mathbf{C_2}$: $\mathbf{C_1}$ contains $\mathbf{C_2}$, $\mathbf{C_2}$ contains $\mathbf{C_1}$, or neither $\mathbf{C_1}$ nor $\mathbf{C_2}$ is contained within the other, as shown in Figure 3.7(a)-(c). For $\mathbf{C_1}$ to contain $\mathbf{C_2}$, a ray from any point on $\mathbf{C_2}$ must intersect $\mathbf{C_1}$ an odd number of times (Figure 3.8(a)) as long as the intersection point(s) is not a tangent. For $\mathbf{C_2}$ to contain $\mathbf{C_1}$, a ray from any point on $\mathbf{C_1}$ must intersect $\mathbf{C_2}$ an odd number of times (Figure 3.8(b)) as long as the intersection point(s) is not a tangent.

For neither $\mathbf{C_1}$ nor $\mathbf{C_2}$ to contain each other, a ray from either curve must intersect the other curve an even number of time (Figure 3.8(c)) as long as the intersection point(s) is not a tangent. If tangents are found, a new ray is needed. This function takes as input a trim loop to be placed within the hierarchy and the hierarchy built so far. Pseudo code for this function is as follows:

Insert (trim loop C_{TL}, trim loop list C_{TLL})
  for *each trim loop C_{TLi} in C_{TLL}* ***do***
    if *C_{TLi} contains C_{TL}* ***then***
      Insert (*C_{TL}, C_{TLi} trim loop list*)
      return
    else if *C_{TL} contains C_{TLi}* ***then***
      Insert (*C_{TLi}, C_{TL} trim loop list*)
      remove C_{TLi} from the C_{TLL} trim loop list
  add C_{TL} to the C_{TLL} trim loop list

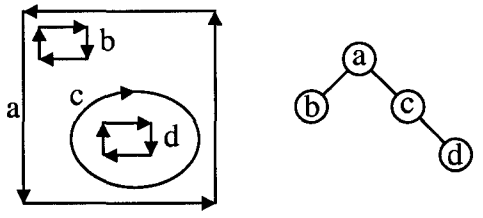


Figure 3.6 Trim curve loops and the resulting hierarchy [57].

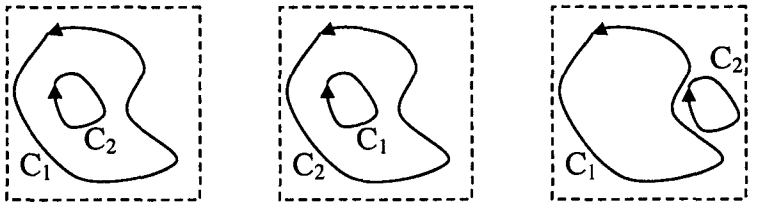
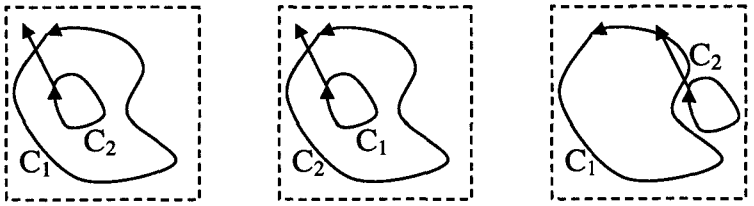


Figure 3.7 (a) C_1 contains C_2, (b) C_2 contains C_1, and (c) C_1 and C_2 are disjoint [57].



(a) (b) (c)

Figure 3.8 Relationship between two trim curve loops.

3.1.12. Orient

The orient function makes sure that all the trim loops are pointing in the appropriate direction. This is simple when using the hierarchy tree of trim loops. The trim loop represented by the root node of the hierarchy tree is always in counter-clockwise direction. Each subsequent level down the hierarchy tree is a trim loop going in the opposite direction. Therefore the trim loops alternate going from counter-clockwise to clockwise in successive levels of the tree. All trim curves on the same level within the tree are oriented in the same direction. The input to this function is the trim loops. Pseudo code for this function is as follows:

```
Orient (trim loops C_TL)
    create an trim loop list C_TLL
    for each trim loop C_TLi in C_TL do
        Insert (C_TLi, C_TLL)
    for each level in the hierarchy tree C_TLL do
        if level is odd then
            orient trim loop counter-clockwise
        else
            orient trim loop clockwise
```

3.2. Loop Closure Methodology

This section provides the details of the three main parts of the loop closure algorithm: the pre-processor, the processor, and the post-processor. Figure 3.9 shows how the functions mentioned above support the loop closure algorithm.

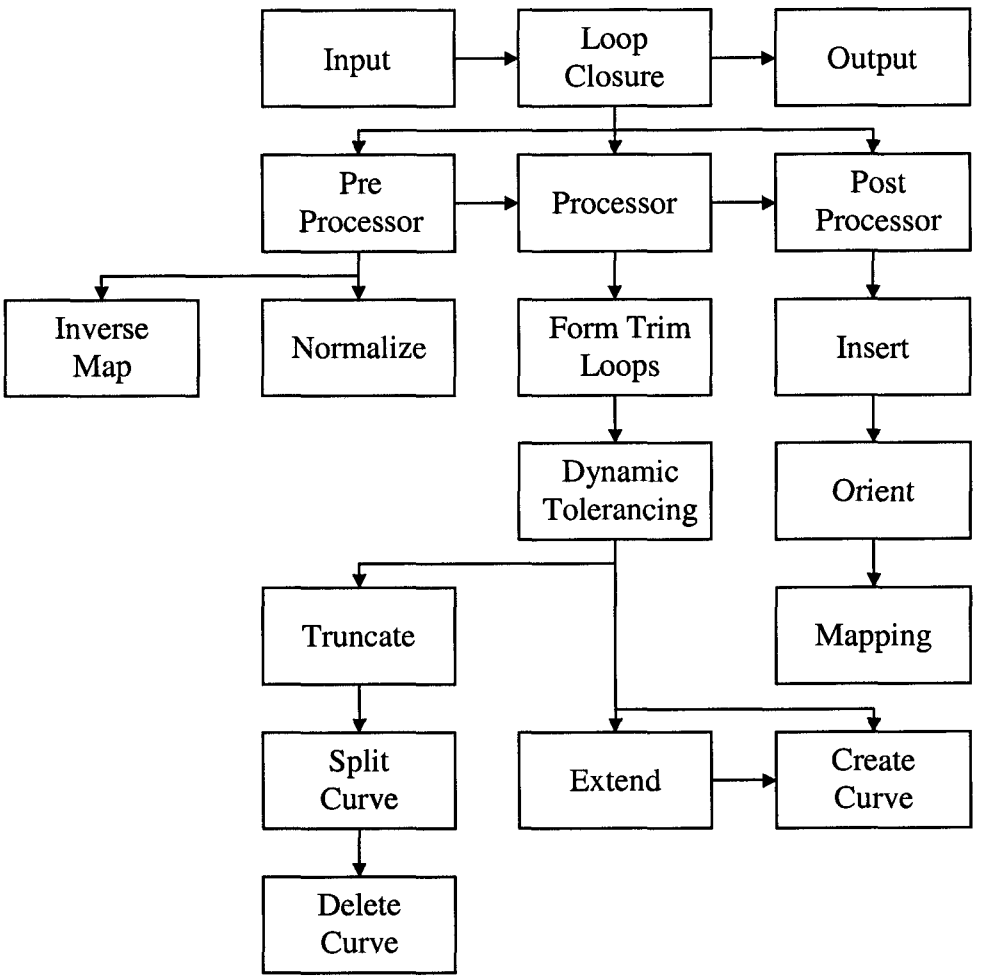


Figure 3.9 The loop closure structure.

3.2.1. Pre-Processor

The pre-processor is the first part of the loop closure algorithm. Its two steps involve getting the geometric data definitions into a form that the loop closure algorithm uses by making sure all trim curves are defined in parameter space and normalizing all NURBS definitions. Making sure everything is defined in parameter space is the first pre-step. If there are only model space curves defined, the corresponding parameter space curves are created using the Inverse Mapping function. The next step is to make sure that the surface and all curves are normalized to a parametric range of [0,1] using the Normalize function. After these two steps, the NURBS definitions of the surface and curves are in their default state. With the proper form for the definitions, the data is ready to proceed to the main processor.

3.2.2. Processor

The processor, the heart of the loop closure algorithm, computes, modifies, and completes trim loops. The functions involved with the processor are Form Trim Loops, Truncate, Expand, Create Curve, Delete Curve, Split Curve, Mapping, and Nearest Neighbor.

The Form Trim Loops function automatically assembles successive trim curves together to form composite trim curves and, if possible (and preferable), trim loops. After the composite trim curves and trim loops have been constructed, the Nearest Neighbor function is used to determine which disjoint trim curves and/or composite trim curves need to be connected to form potential trim loops. The trim loops do not need to be evaluated since they already form a closed loop.

The next step is to automatically detect the reason why a trim curve or composite trim curve is invalid and correct the problem. There are three possible reasons why trim curves are not closed: a small gap exists between endpoints of trim curves, an open section exists where a trim curve needs to be created, or a catastrophic unsolvable error exists. Of these three errors, the first two are correctable by the loop closure algorithm. The unsolvable errors are caused by the authoring CAD system and can only be corrected from within.

The gap problem has two solutions, either finding an intersection of the trim curves or joining the trim curves with a linear segment if no intersection exists. If the trim curves already have an intersection point that is not their endpoint, the Truncate function is used to remove the overhang. If the trim curves do not already have an intersection, the Extend function is used to see if a valid one exists. If a valid intersection cannot be computed, then the Create Curve function is used to create a line segment between the two trim curve endpoints.

Four cases exist that encompass the problem areas associated with trim curve error resolution: no trim curves, one (composite) trim curve, two (composite) trim curves, and more than two (composite) trim curves.

3.2.2.1. No Trim Curves

The first trivial case is when there are no parameter space trim curves available. This also means there are no model space trim curves either in which to produce the parameter space curves. With this case, it is assumed that the entire surface is needed. Therefore, a trim loop is defined that encloses all of parameter space. Four individual trim curves are

defined on each of the parameter space boundaries (Figure 3.10) using the Create Curve function.

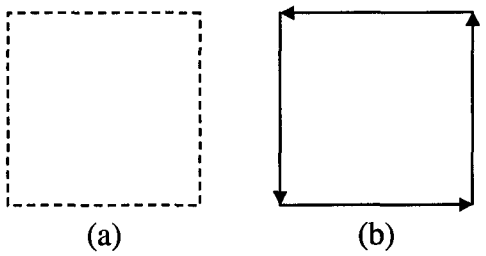


Figure 3.10 (a) No trim loops and (b) complete boundary trim loop.

3.2.2.2. One Trim Curve

The second case of trim curve error resolution is when there is only one trim curve. There are three positions in parameter space that this curve can occupy: the endpoints are in the interior (Figure 3.11(a)), one endpoint is on the boundary and the other in the interior (Figure 3.11(b)), or both endpoints are on the boundary (Figure 3.11(c)).

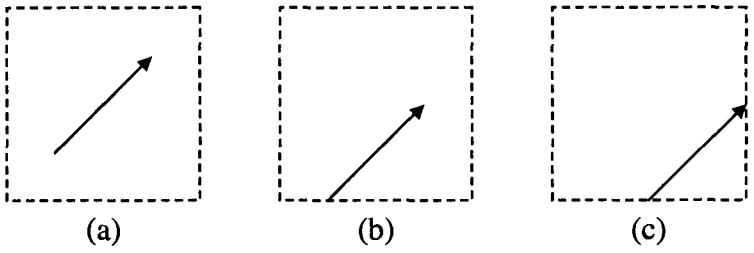


Figure 3.11 Spatial positions of one trim curve in parameter space.

For the first two conditions, there are two situations to address, that the trim curve may or may not be a linear segment as shown in Figure 3.12(a)-(d). If the trim curve is linear, there is not enough information to create a trim loop causing an unsolvable error. If the trim curve

is not linear, then the Truncate, Extend, or Create Curve functions are used to connect the endpoints as long as an intersection, other than at an endpoint, is not created.

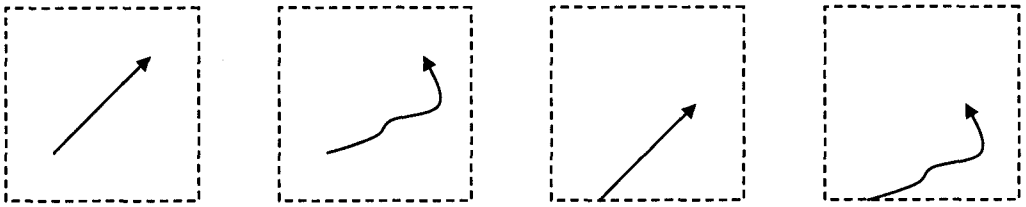


Figure 3.12 Linear and non-linear interior and semi-interior trim curves.

When both endpoints are on the boundary, traversing around the boundary in a counter-clockwise direction from the head to the tail of the trim curve defines the trim loop (Figure 3.13). An area of concern is that the boundary curves that are created cannot intersect the trim curve at more than point contact intersections. If this condition is detected, the loop cannot be resolved and a fatal error is reported.

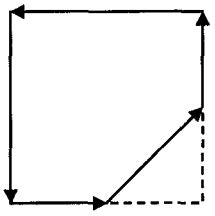


Figure 3.13 Completed boundary trim loop.

3.2.2.3. Two Trim Curves

The third case is when there are two trim curves. The Dynamic Tolerancing function determines if these trim curves should be joined. If so, the Truncate, Extend, or Create Curve function is used to connect them. If not, each one is evaluated independently using the one trim curve case.

3.2.2.4. More Than Two Trim Curves

The fourth case is when there are more than two trim curves. Again the Dynamic Tolerancing function determines which trim curves should be joined. For each pair of joinable trim curves, the two trim curve case is recursively used until the one trim curve case can be used for completion or until failure.

3.2.3. Post-Processor

The final step in processing a face is to ensure that each trim loop is properly oriented, counter-clockwise for regions or clockwise for holes. This is the final step to ensure that a robust definition of a trimmed surface exists.

The first step is to ensure that the individual trim curves within a trim loop are consistently oriented, as shown in Figure 3.14. This should have already been done by passing through the forming trim loops phase, which connects the curves head to tail, but one last check is done for robustness.

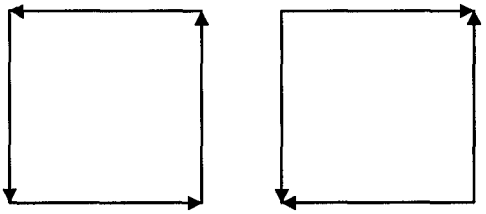


Figure 3.14 (a) Proper vs. (b) improper orientation of trim curves within a trim loop.

After all the trim loops are consistently oriented, the second step is to build the containment-based hierarchy using the list of trim loops and the Insert function. After the hierarchy is created, orienting the trim curves becomes trivial. The root of the hierarchy tree

is the outer most trim loop and must, by definition, by oriented in a counter-clockwise direction. Each subsequent progression down the tree orients the trim loop in the opposite direction, alternating between clockwise and counter-clockwise directions.

Once full definition of the parameter space trim curves is attained, any mappings to model space are done.

Chapter 4. Examples

The loop closure methodology was implemented using International TechneGroup Incorporated's (ITI's) PDElib [58] to parse the IGES and STEP files and filter the geometric data. The algorithm was written in C++ and operates on various platforms.

This chapter presents examples of IGES and STEP files created using a variety of commercial CAD systems. These files contain a variety of errors that are representative of the current state of the art. In most cases, the neutral file can be read successfully by the system that created it, since the same assumptions used in writing the data are generally employed in reading and interpreting it. However, most of the examples exhibited fundamental problems when read by a system other than the authoring system. Typical errors encountered include improperly addressing poles, seams, and degenerate edges (Appendix E) and gaps between trim curves.

In each example, the model is first shown using a visualization tool that requires complete closed loops (in both parameter space and model space) in order to robustly tessellate and display the models. After the loop closure algorithm is applied to the models they are shown with the same visualization system.

While these types of errors are demonstrated with a visualization application, they are representative of the types of problems encountered by many downstream analysis systems.

4.1. Pole Errors

Figure 4.1 shows five primitives: a torus, a sphere, a cone, a cylinder, and a cube. The IGES data file was created using I-DEAS Master Series 5 and saved using its IGES exporter.

This collection of primitives contains 19 faces, of which four need correction using the loop closure algorithm. The torus, cylinder, and cube provide complete surface, model space trim curve, and parameter space trim curve information, but the sphere and cone do not. Both of these primitives have the same problem; they have poles that correspond to trim curves and need to be defined in parameter space and then mapped to their corresponding singular point in model space. Since the sphere, cone, and cylinder were defined as two halves and the torus as four sections, potential seam problems were avoided. The cube, defined as six planar faces, has neither a pole nor seam problem.

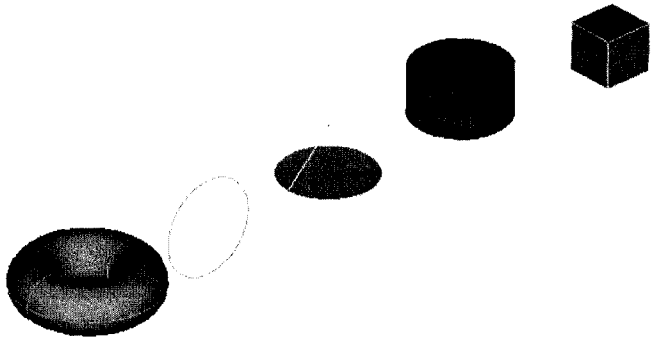


Figure 4.1 Primitives with pole errors.

Figure 4.2(a) and (b) show the incomplete parameter space definitions for the pole problem of the cone and Figure 4.2(c) and (d) show it for the sphere. The open segments in parameter space indicate the location of the poles. The cone composite trim curve consists of three connected parameter space curves (*a*, *b*, and *c*). The sphere trim curves consist of two unconnected parameter space curves (*a* and *b*). In model space, the curves appear connected (Figure 4.3(a)-(d)).

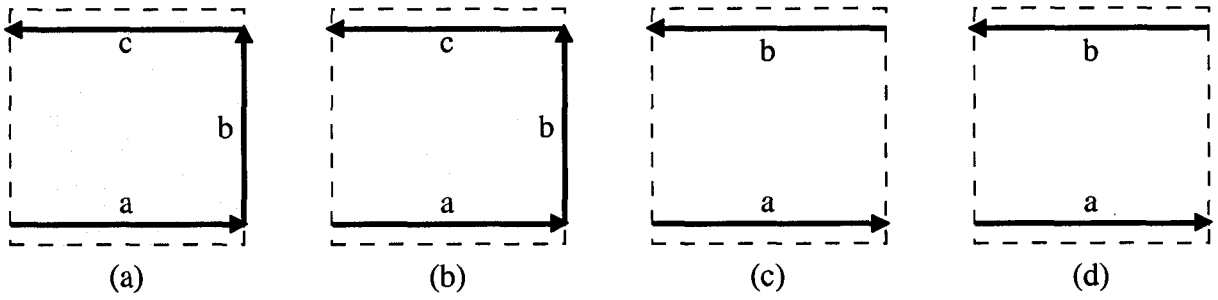


Figure 4.2 Parameter space trim curves exhibiting a pole error.

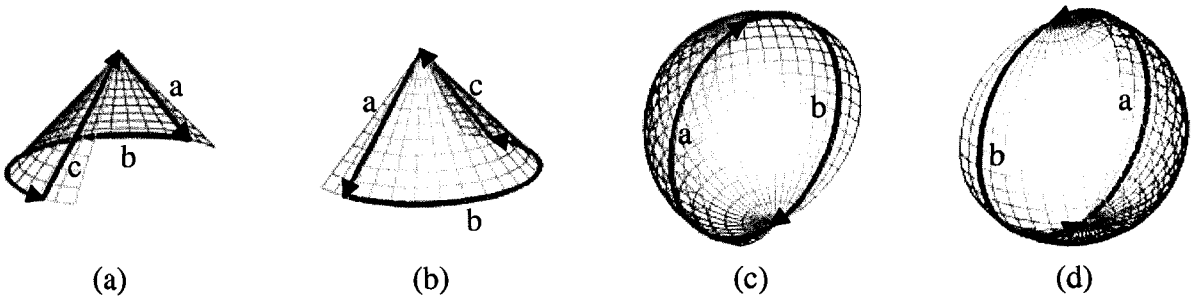


Figure 4.3 Model space trim curves exhibiting a pole error.

Figure 4.4(a)-(d) and Figure 4.5(a)-(d) show the complete parameter space and model space definitions for the conical and spherical surfaces after having been completed by the loop closure algorithm. Each trim loop now consists of four connected parameter space curves (*a*, *b*, *c*, and *d*) that trim the underlying parent surfaces to provide the necessary half surfaces of the cone and sphere that are needed.

Figure 4.6 shows the same five primitives after having passed through the loop closure algorithm. The torus, cylinder, and cube needed no correction and were bypassed by the correction routine, but the cone and sphere were captured and corrected.

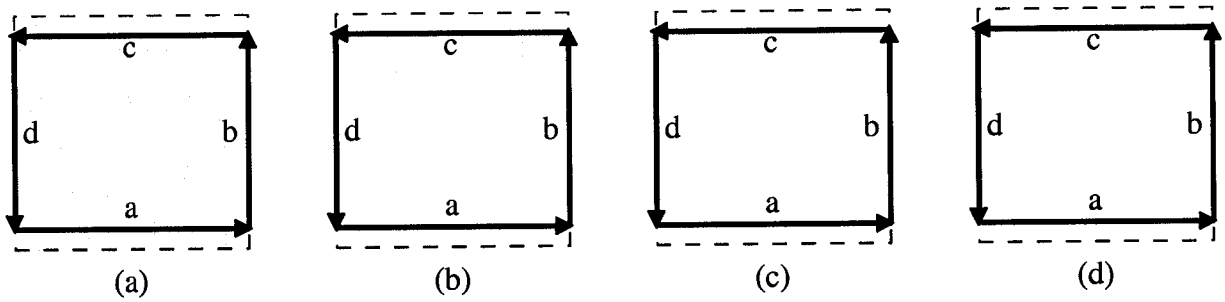


Figure 4.4 Parameter space trim curves without a pole error.

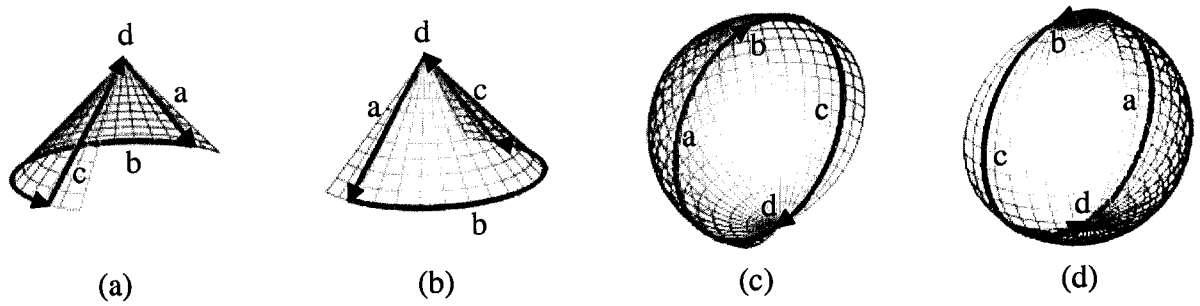


Figure 4.5 Model space trim curves without a pole error.

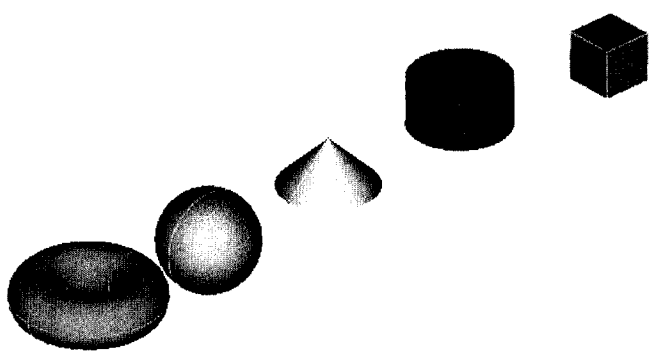


Figure 4.6 Primitives without pole errors after loop closure.

4.2. Gap Errors

Figure 4.7 shows a simple motor housing unit created using CATIA. This STEP file contains 23 faces with three needing repair. Two of these faces contain gaps between parameter space curves most likely caused by a machine precision problem, or they may have been slightly ill defined.

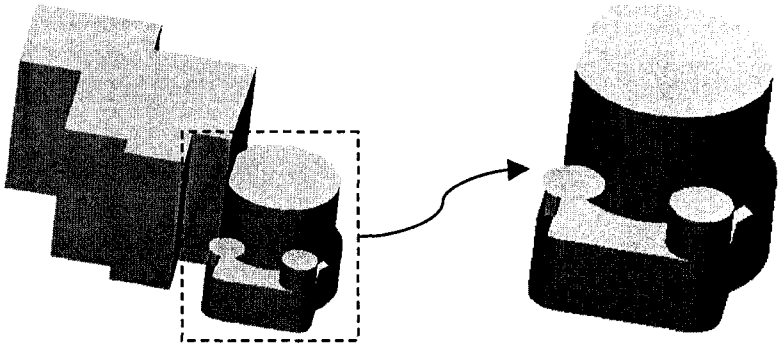


Figure 4.7 Motor housing unit with gap errors.

Figure 4.8(a) shows the parameter space curves for a face containing a gap. The blow-ups are of the gap area (Figure 4.8(b)) and a section containing a very short parameter space curve (Figure 4.8(c)), perhaps an area where a gap was detected and corrected by the native CAD package. Figure 4.9(a)-(c) shows the corresponding model space curves. The composite trim curve for this surface consists of seven curves (*a*, *b*, *c*, *d*, *e*, *f*, and *g*).

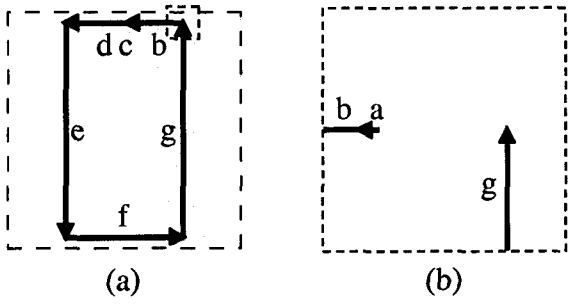


Figure 4.8 Parameter space trim curves exhibiting a gap error.

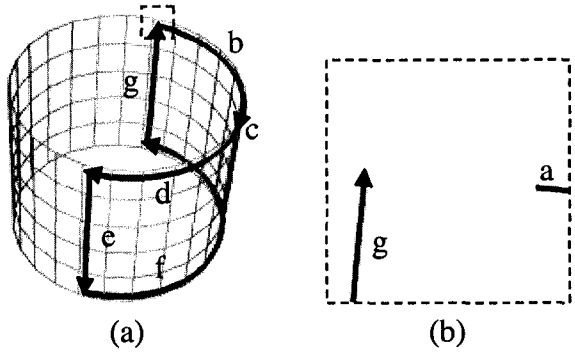


Figure 4.9 Model space trim curves exhibiting a gap error.

Figure 4.10(a)-(c) and Figure 4.11(a)-(c) show the complete parameter space and model space definition after having passed through the loop closure algorithm. Computing the common intersection point of parameter space curves *a* and *g* closes the gap (Figure 4.10(b)).

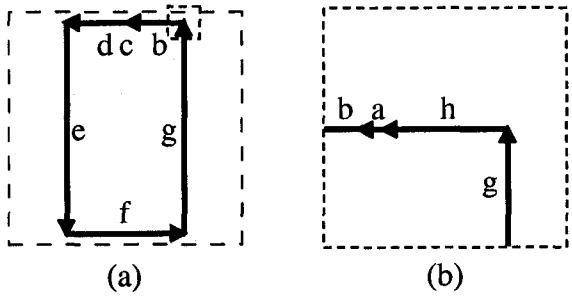


Figure 4.10 Parameter space trim curves without a gap error.

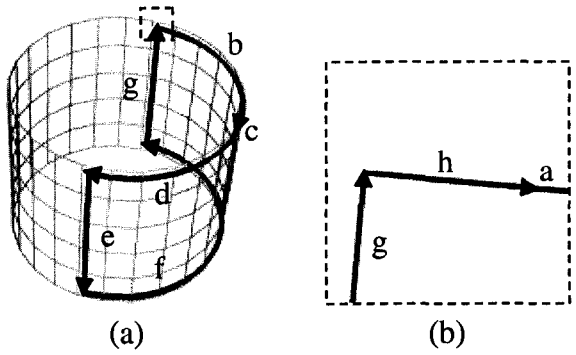


Figure 4.11 Model space trim curves without a gap error.

Figure 4.12 shows the motor housing unit after having passed through the loop closure algorithm.

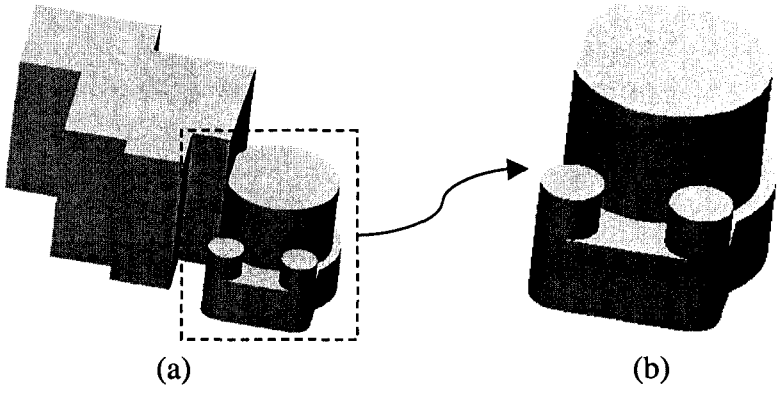


Figure 4.12 Motor housing unit without gap errors after loop closure.

4.3. Seam Errors

Figure 4.13 shows a sheet metal part with many cylindrical holes. This STEP file was created using Unigraphics 15.0 and contains 107 faces with 15 of them needing repair. All 15 faces are identical and contain seam problems.

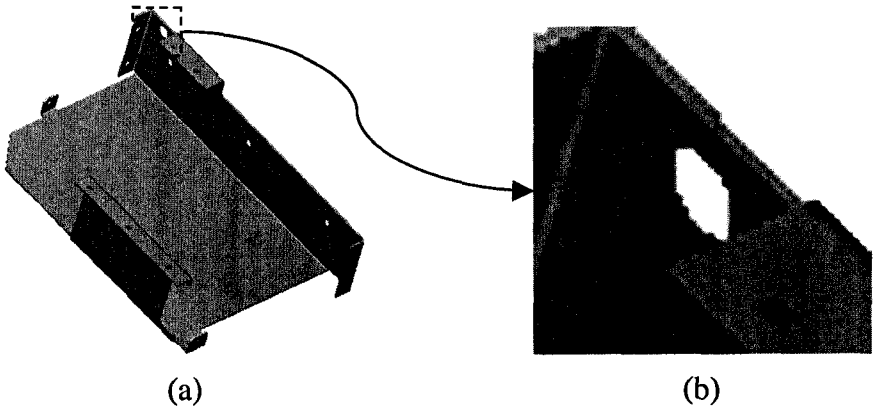


Figure 4.13 Sheet metal part with seam errors.

Figure 4.14 shows the incomplete parameter space curves for one of the faces with a seam problem. The trimmed surface contains one composite trim curve made up of five trim curves (*a*, *b*, *c*, *d*, and *e*) with an open segment where the seam occurs in model space running along curve *c* (Figure 4.15). A possible reason for two curves defining a single edge, such as [*a*,*b*] or [*d*,*e*], is to alleviate another seam problem that was rotated out-of-the-way by 180 degrees. However, this seems unlikely in this case because a seam problem still exists.

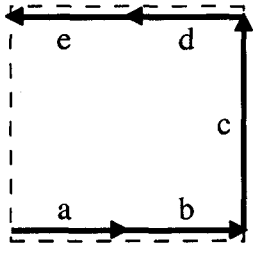


Figure 4.14 Parameter space trim curves exhibiting a seam error.

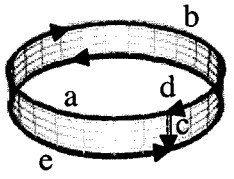


Figure 4.15 Model space trim curves exhibiting a seam error.

Figure 4.16 and Figure 4.17 show the complete parameter space and model space curve definitions for this face after having been completed by the loop closure algorithm. The trim loop now consists of six trim curves (*a*, *b*, *c*, *d*, *e*, and *f*) that trim off a sliver of the upper and lower portion of the cylindrical surface.

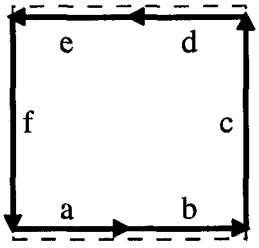


Figure 4.16 Parameter space trim curves without a seam error.

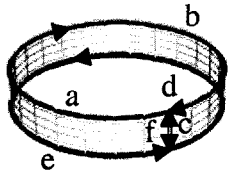
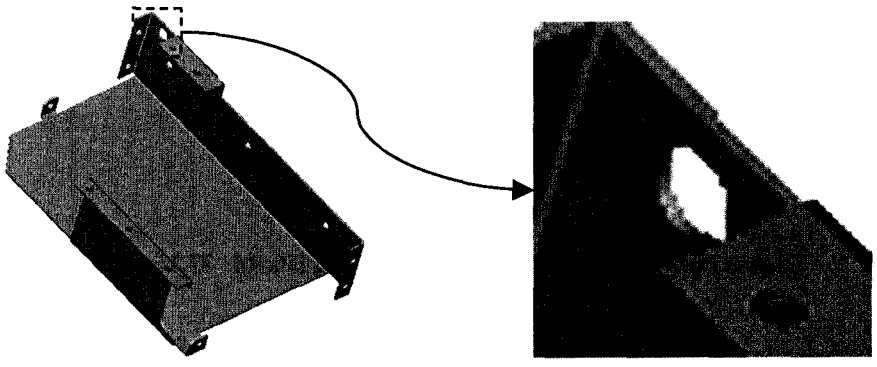


Figure 4.17 Model space trim curves without a seam error.

Figure 4.18 shows the sheet metal part after having been corrected with the loop closure methodology. The cylindrical faces now appear within the 15 holes made in the sheet metal part.



(a) (b)

Figure 4.18 Sheet metal part without seam errors after loop closure.

4.4. Degenerate Edge Errors

Figure 4.19 shows a converter surface created using PDGS Version 26.03 and exported as an IGES file. This model contains 140 faces of which 29 need repair. Several pole and degenerate edge problems exist within this model, but only the 14 identical degenerate edges are addressed here since the other 15 are not degenerate edges.

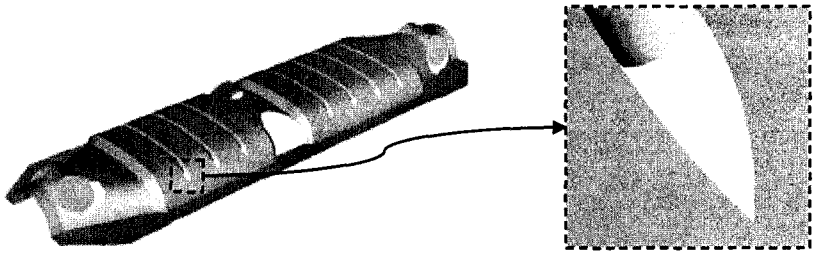


Figure 4.19 Converter surface with degenerate edge errors.

Figure 4.20 shows the parameter space curves for one of the faces containing a degenerate edge. This trimmed surface contains one composite trim curve comprised of four trim curves (*a*, *b*, *c*, and *d*) with an open section where the degenerate edge is located in

model space (Figure 4.21). The lower parameter edge is comprised of two edges instead of one, possibly to alleviate a seam problem that was rotated out-of-the-way by 180 degrees.

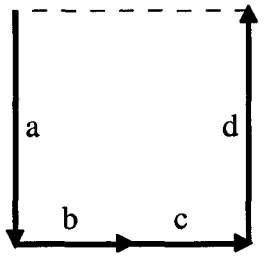


Figure 4.20 Parameter space trim curves exhibiting a degenerate edge error.

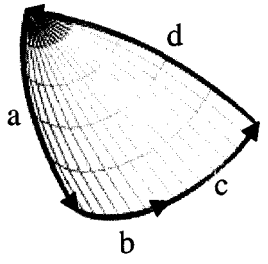


Figure 4.21 Model space trim curves exhibiting a degenerate edge error.

Figure 4.22 shows the complete parameter space curves for the face containing a degenerate edge. The trim loop now consists of five trim curves (*a*, *b*, *c*, *d*, and *e*). Since all the parameter space curves are entirely enclosed on the boundary of parameter space, the entire underlying surface is trimmed and kept in model space (Figure 4.23).

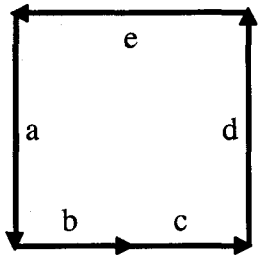


Figure 4.22 Parameter space trim curves without a degenerate edge error.

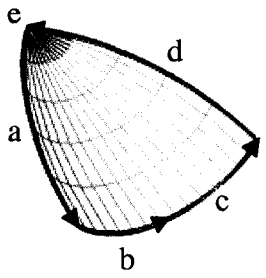


Figure 4.23 Model space trim curves without a degenerate edge error.

Figure 4.24 shows the converter surface after having passed through the loop closure. Although this model looks completely correct, there are still three surfaces that are not correct. These limitations are addressed further in the conclusion.

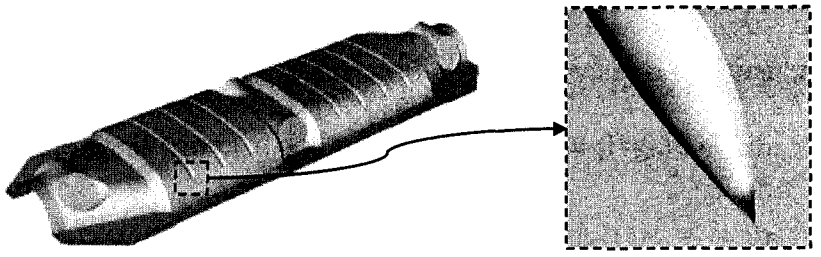


Figure 4.24 Converter surface without degenerate edge errors after loop closure.

Chapter 5. Conclusions

This dissertation introduces the loop closure methodology for providing a robust definition of parameter space and model space trim curves for a given surface. The goal is to improve the reliability of data transfer between different CAD/CAM/CAE systems as well as other engineering applications, analysis tools, and visualization software using the neutral file formats IGES and STEP.

This method demonstrates a marked improvement of data transfer over the default translators and exporters provided by vendors. Of the 24 files of various complexities tested that needed correction, 22 of them were corrected completely, for a 91.7% success rate. There were a total of 10,229 faces within the 24 files of which 330, or 3.22%, that needing repair or completion. Of these 330 faces, 326 of them were properly corrected, for a success rate of 98.8%, leaving only 0.0391% of the total number of faces still with errors.

Table 5.1 lists the test files used along with their file type and the CAD package that was used to create them. From this list, it is evident that many different CAD packages suffer from similar poor definitions of their models. Table 5.2 shows the statistics that were gathered testing these files. However, this only represents the files that had trim curve problems, many files were tested that did not contain any errors.

Table 5.1 Test files.

File #	Model Name	CAD Package	File Type
1	Block Assembly	I-DEAS Master Series 5	IGES
2	Converter Surface	PDGS Version 26.03	IGES
3	Cap	Pro/E	IGES
4	Head	Pro/E	IGES
5	Push Rod	Pro/E	IGES
6	Manifold	I-DEAS Master Series 7	IGES
7	Primitives	I-DEAS Master Series 5	IGES
8	Screw	CoCreate SolidDesigner	IGES
9	Mounting (AP203)	Unigraphics 14.0	STEP
10	Mounting (AP214)	Unigraphics 14.0	STEP
11	Fender #1	Pro/E	STEP
12	Fender #2	Pro/E	STEP
13	Fender #3	Pro/E	STEP
14	Screw	CoCreate SolidDesigner	STEP
15	Chassis	CoCreate SolidDesigner	STEP
16	Housing Printed Wiring Board	I-DEAS Master Series 5	STEP
17	Key Extension Switch	I-DEAS Master Series 5	STEP
18	Control Switch	I-DEAS Master Series 5	STEP
19	Pipe	Pro/E	STEP
20	Cooling System	CATIA	STEP
21	Sheet Metal	Unigraphics 15.0	STEP
22	Wheel Assembly	EAI STEP Exporter 1.0	STEP
23	Battery Case	Pro/E	STEP
24	Motor Housing	CATIA	STEP

Table 5.2 Test file statistics.

File #	# of Parts	# of Faces	# of Bad Faces	% Bad Faces	# of Repaired Faces	% of Repaired Faces
1	1	32	1	3.1	1	100.0
2	1	140	29	20.7	26	89.7
3	1	17	2	11.8	2	100.0
4	1	108	2	1.9	2	100.0
5	1	9	4	44.4	4	100.0
6	1	50	1	2.0	1	100.0
7	1	19	4	21.1	4	100.0
8	1	139	2	1.4	2	100.0
9	1	341	78	22.9	78	100.0
10	1	341	78	22.9	78	100.0
11	1	163	1	0.6	1	100.0
12	1	371	1	0.3	1	100.0
13	1	108	3	2.8	3	100.0
14	1	123	2	1.6	2	100.0
15	12	5642	64	1.1	64	100.0
16	1	330	18	5.5	18	100.0
17	1	70	4	5.7	4	100.0
18	1	43	4	9.3	4	100.0
19	1	79	4	5.1	4	100.0
20	89	1075	4	0.4	4	100.0
21	1	107	15	14.0	15	100.0
22	27	661	4	0.6	4	100.0
23	5	238	2	0.8	2	100.0
24	3	23	3	13.0	2	66.7

From the tables, it is evident that the loop closure algorithm did not solve every problem, although it did solve most of them. Figure 5.1(a)-(d) shows the parameter space curves for the three faces of the converter surface and one face of the motor housing unit that were not repairable by the loop closure algorithm. Figure 5.1(a) and (c) show significant portions of trim curves that lay directly on each other causing multiple intersection points, Figure 5.1(b) shows a situation where the loop closure algorithm would cause an intersection point violating the criteria for a correct trim curve, and Figure 5.1(d) is a situation of a single linear curve with no section touching the parameter space boundary. As stated in Chapter 3, the loop closure bypasses this situation as unsolvable. Figure 5.2 shows the corresponding model space curves and underlying parent surface. In model space, the curves appear relatively accurate. Therefore, the problems associated with these faces are caused by catastrophic errors, most likely within the originating CAD system.

However, based on manual observation of these unsolvable faces when compared with the other faces within the models, there are simple solutions to correcting them. The existing parameter space curves of Figure 5.1(a)-(c) need to be discarded and then a boundary trim loop created trimming the entire parent surface. This seams appropriate since other similar faces exist in the model that were provided correctly or were corrected with the loop closure algorithm. The lone parameter space curve shown in Figure 5.1(d) is considered a machine precision error because of its small magnitude and, therefore, can be discard completely. Although simple solutions exist to correcting these four problems, they are solved through manual (interactive) observation, violating the desirable automatic correction algorithm.

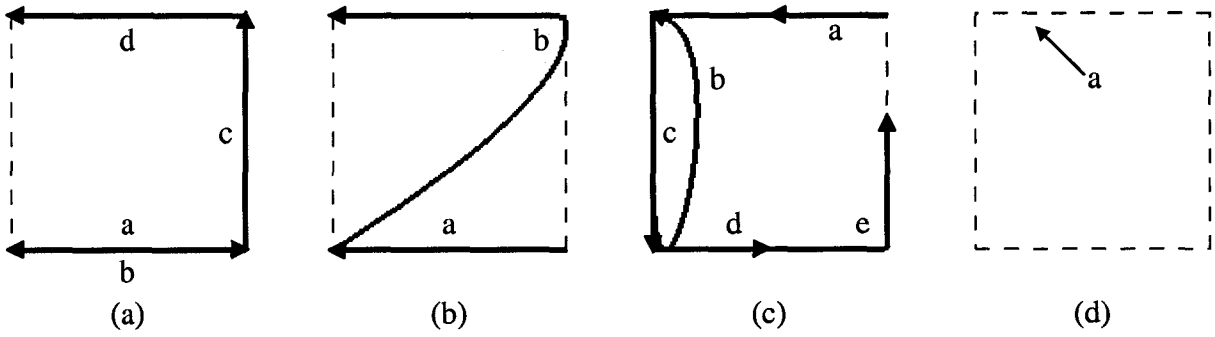


Figure 5.1 Parameter space curves for unsolved faces.

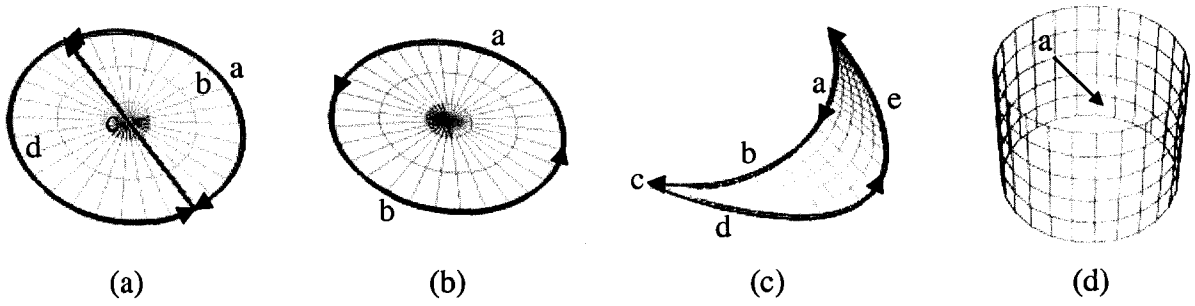


Figure 5.2 Model space curves for unsolved faces.

There have been and still are problems occurring when a particular system's translator is interfaced with IGES or STEP. Such problems are attributed to several factors: different interpretation of the specification leading to poor interoperability, different geometric tolerances, vendors' low priority commitment to interoperability, modeling practice variations, degeneration of information, corruption of files, differences in modeling standards used within companies, manual data entry, and inverse relationship between model complexity and translator reliability. However, the area of "CAD repair" has gained significant momentum as the benefits of fast, consistent interoperability between systems gains more recognition.

Recently, several research and corporate solutions have emerged. The research movement has focused on two major approaches to geometric correction: correction of polygonal descriptions of the models by addition of polygons or different stitching operations [49] [50] [51] [52] [53] [54] and correction of the exact geometric definition by modifying the topological definition of the model [47] [48] [59] [60] [61] [62]. The loop closure algorithm follows the second approach. The corporate solutions include ITI's CADfix [63], XOX Corporation's GDX [64], Avatech Solutions' DesignQA [65] and GeometryQA [66], and TransMagic, Inc.'s translation software products [67].

With the growing abundance of research and corporate solutions addressing the CAD repair problem, the relevance of this dissertation as a step toward the complete and accurate representation of geometric CAD data is evident. Even though CAD vendors have progressed toward more reliable solutions themselves, they are still lacking a 100% success rate. The future of this work would be to directly interface the loop closure methodology as a check for complete model representation within a CAD system. Having CAD systems that produce completely error free models would alleviate the need for downstream applications wasting the time of checking and correcting the models. This would save users much needed time, money, and headaches of dealing with poorly defined CAD models.

Appendix A. NURBS

NURBS are the *de facto* industry standard for the representation, design and data exchange of geometric information processed by computers. Thus, several national and international standards, such as IGES and STEP, have incorporated its use within their specification.

NURBS map independent parameters from one domain to dependent variables of another domain. For example, a 3D NURBS curve maps a single independent parameter u to the dependent variables x, y, and z; and a 3D NURBS surface maps two independent parameters u and v to the dependent variables x, y, and z.

The following is a brief introduction to the NURBS concepts used within this research work. For a more complete presentation, refer to *The NURBS Book* [19].

A.1. NURBS Curve

A NURBS curve is defined by the equation

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad a \le u \le b. \tag{A.1}$$

where u is the independent parameter defined on the interval $[a,b]$, p is the degree, $\{\mathbf{P}_i\}$ are the *control points* forming a *control polygon*, $n+1$ is the number of control points, $\{w_i\}$ are the *weights*, and $\{N_{i,p}(u)\}$ are the basis functions with the parametric range defined by the *knot vector*

$$U = \{\underbrace{a,\ldots,a}_{p+1}, u_{p+1}, \ldots, u_{m-p-1}, \underbrace{b,\ldots,b}_{p+1}\}$$

where $m = n + p + 1$ are the number of knots and $\{u_i\}$ for $i = 0,\ldots,m-1$ are the knots. The $p+1$ repeated end knots make the knot vector *non-periodic* and the interior knots $\{u_{p+1},\ldots,u_{m-p-1}\}$ do not have to be evenly spaced making the knot vector *non-uniform*. The knot vector is a non-decreasing sequence of real numbers, $u_i \le u_{i+1}$ for $i = 0,\ldots,m-1$. An example of a NURBS curve with differing weights is shown in Figure A.1.

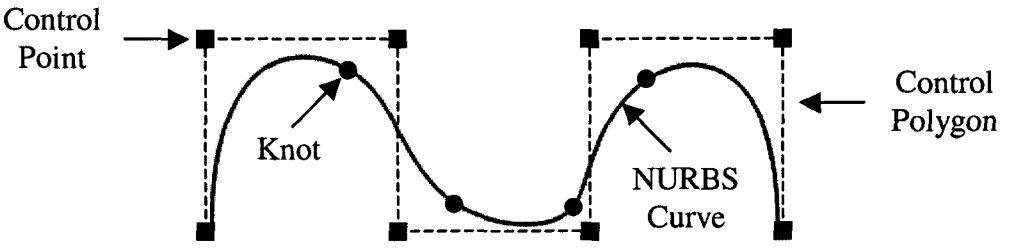


Figure A.1 A NURBS curve.

A.2. NURBS Surface

A NURBS surface is defined by the equation

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad a \le u \le b, c \le v \le d \tag{A.2}$$

where u and v are the independent parameters defined on the intervals $[a,b]$ and $[c,d]$, p is the degree in the u-direction, q is the degree in the v-direction, $n+1$ is the number of control points in the u-direction, $m+1$ is the number of control points in the v-direction, $\{\mathbf{P}_{i,j}\}$ are the *control points* forming a bi-directional *control net*, $\{w_{i,j}\}$ are the *weights*, $\{N_{i,p}(u)\}$ are

the basis functions in the u-direction and $\{N_{j,q}(u)\}$ are the basis functions in the v-direction defined on the non-periodic, non-uniform, non-decreasing knot vectors

$$U = \{\underbrace{a,\ldots,a}_{p+1}, u_{p+1},\ldots,u_{r-p-1}, \underbrace{b,\ldots,b}_{p+1}\}$$

$$V = \{\underbrace{c,\ldots,c}_{q+1}, v_{q+1},\ldots,v_{s-q-1}, \underbrace{d,\ldots,d}_{q+1}\}$$

where $r = n + p + 1$ are the number of knots and $\{u_i\}$ for $i = 0,\ldots,r-1$ are the knots in the u-direction and $s = m + q + 1$ are the number of knots and $\{v_j\}$ for $j = 0,\ldots,s-1$ are the knots in the v-direction. An example of a NURBS surface is shown in Figure A.2.

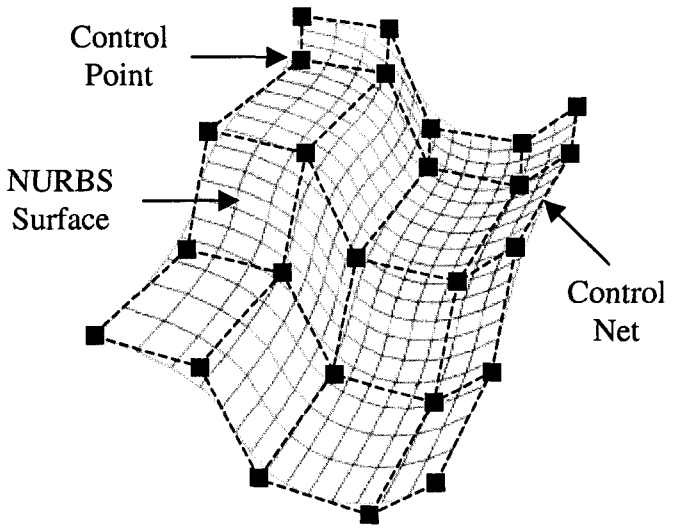


Figure A.2 A NURBS surface.

A.3. NURBS Curve Derivatives

The derivative of a NURBS curve is defined by the equation

$$\mathbf{C}^{(k)}(u) = \frac{\mathbf{A}^{(k)}(u) - \sum_{i=1}^{k} \binom{k}{i} w^{i}(u)\mathbf{C}^{(k-i)}(u)}{w(u)} \qquad \text{(A.3)}$$

where $\mathbf{C}^{(k)}(u)$ and $\mathbf{C}^{(k-i)}(u)$ are the k^{th} and $(k\text{-}i)^{th}$ derivatives of $\mathbf{C}(u)$, $\mathbf{A}^{(k)}(u)$ is the k^{th} derivative of $\mathbf{A}(u)$, and $w^{(i)}(u)$ is the i^{th} derivative of $w(u)$. So, $\mathbf{C}^1(u)$ is the first derivative of $\mathbf{C}(u)$, meaning slope continuity, and $\mathbf{C}^2(u)$ is the second derivative of $\mathbf{C}(u)$, meaning curvature continuity. A NURBS curve with the first derivative computed at $u = 0.5$ is shown in Figure A.3.

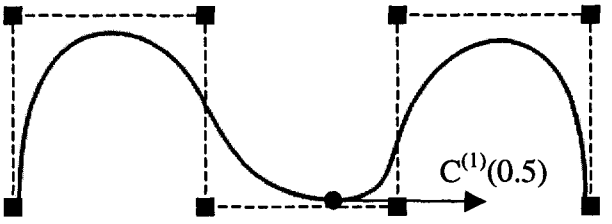


Figure A.3 First derivative of a cubic NURBS curve computed at $u = 0.5$.

A.4. Trim Curves

Surface trim curves are defined in two ways. First they may be defined independently of the surface as 3D curves in the same coordinate system (models space) as the surface itself. Alternatively, they may be defined within the parameter space of the surface (e.g. $\mathbf{C}(t) = \{u(t), v(t)\}$) and require composition with the surface definition for evaluation (i.e. $\mathbf{S}(u(t), v(t))$), as shown in Figure A.4.

Trim curves remove sections from the surface that are not aligned with the underlying parameterization. For example, portions to the left of the trim curve are retained, while those to the right are removed. Using trim curves is a common method for overcoming the topologically rectangular limitations of a NURBS surface.

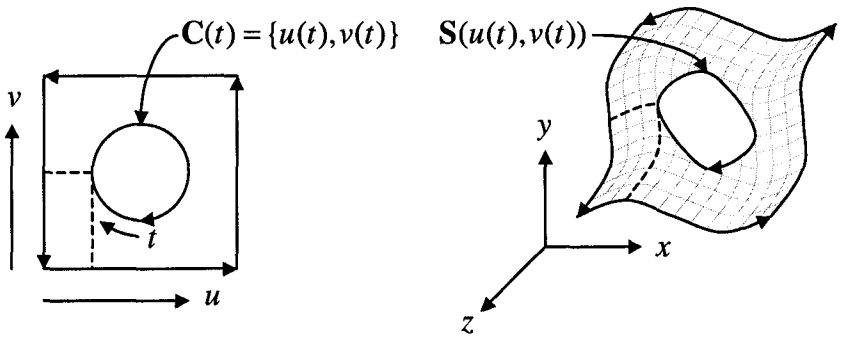


Figure A.4 Parameter space and model space trim curves.

Collections of trim curves form closed trim loop boundaries that partition the NURBS surface. Trim curves may consist of one curve that forms a closed loop on itself or a collection of curves joined together to form a closed loop. In a list of curves forming a closed loop, each curve's last point is coincident with the following curve's first point except for the last curve's last point which is coincident with the first curve's first point.

Trim loops specify the outer-most boundary of a surface as well as any internal holes and regions. The outer-most boundary and internal regions are indicated by counter-clockwise direction trim curves and internal holes are indicated by clockwise direction trim curves. Trim loops may be nested within each other, but each subsequent trim loop must be oriented in the opposite direction, with the outer most trim loop always in the counter-clockwise direction. Trim loops may share vertices and edges, but they are not allowed to cross one another. The entire trim loop must also fall within the parameter space domain of the surface.

A.5. Mapping Parameter Space to Model Space Curves

Mapping from parameter space to model space solves the dependent (x, y, z) values given the independent (u, v) values for a curve lying on a parametric surface. Given a NURBS parameter space curve $\mathbf{C}_p(t) = \{u(t), v(t)\}$ and its parent NURBS surface $\mathbf{S}(u, v)$, the equivalent NURBS model space curve $\mathbf{C}_m(t) = \{x(t), y(t), z(t)\}$ is computed by interpolating surface data points $\mathbf{S}_i(u(t_i), v(t_i))$, where $i = 1 \ldots n$ and n is the number of points such that the model space curve meets a predetermined criteria of closeness to the surface. The $(u(t_i), v(t_i))$ are solved by incrementally moving along the parameter space curve from beginning to end by $\Delta t = 1/(n-1)$, assuming a parametric range of [0,1] for t. The surface data points $\mathbf{S}_i(u(t_i), v(t_i))$ are computed using these parameter values with the surface definition. Figure A.5 illustrates the mapping from parameter to model space.

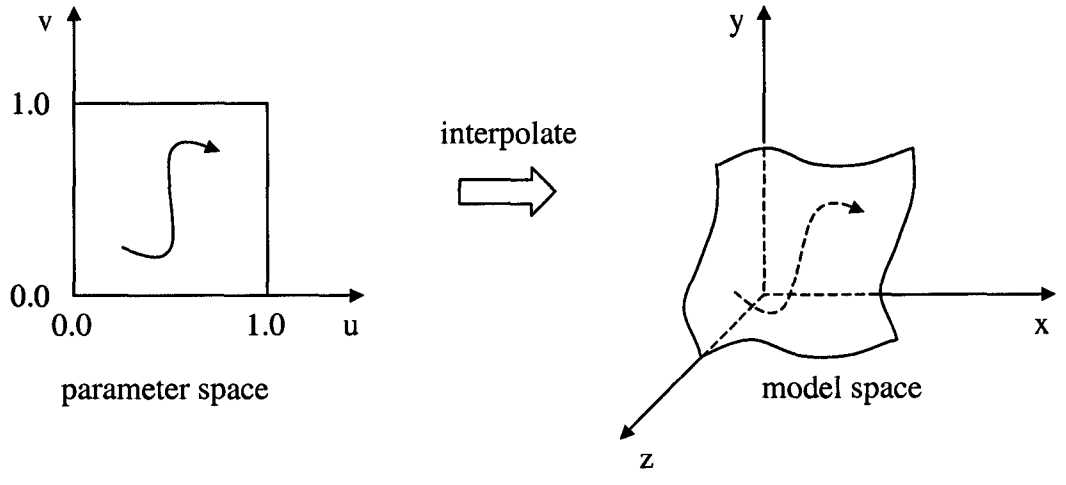


Figure A.5 Mapping parameter space to model space.

A.6. Inverse Mapping Model Space to Parameter Space Curves

Given a NURBS model space curve $\mathbf{C}_m(t) = \{x(t), y(t), z(t)\}$ that lies in the vicinity of a NURBS surface $\mathbf{S}(u,v)$, inverse mapping refers to finding its image $\mathbf{C}_p(t) = \{u(t), v(t)\}$ in parameter space [8] [68]. Iteratively marching along $\mathbf{C}_m(t)$ by $\Delta t = 1/(n-1)$ computes surface data points $\mathbf{S}_i(u(t_i), v(t_i))$, where $i = 1 \ldots n$ and n is the number if data points that produces an accurate representation of the model space curve. The simplest method for solving the inverse mapping problem is to solve the point projection problem using Newton iteration to minimize the distance between the model space curve point $\mathbf{S}_i(u(t_i), v(t_i))$ and the surface point $\mathbf{S}(u,v)$ for some (u,v). Therefore the offset vector function

$$\mathbf{r}(u,v) = \mathbf{S}(u,v) - \mathbf{S}_i(u(t_i), v(t_i))$$

and two angular difference scalar equations

$$f(u,v) = \mathbf{r}(u,v) \cdot \mathbf{S}_u(u,v) = 0$$

$$g(u,v) = \mathbf{r}(u,v) \cdot \mathbf{S}_v(u,v) = 0$$

are solved seeking point coincidence and zero cosine convergence. Figure A.6 illustrates the mapping from model to parameter space.

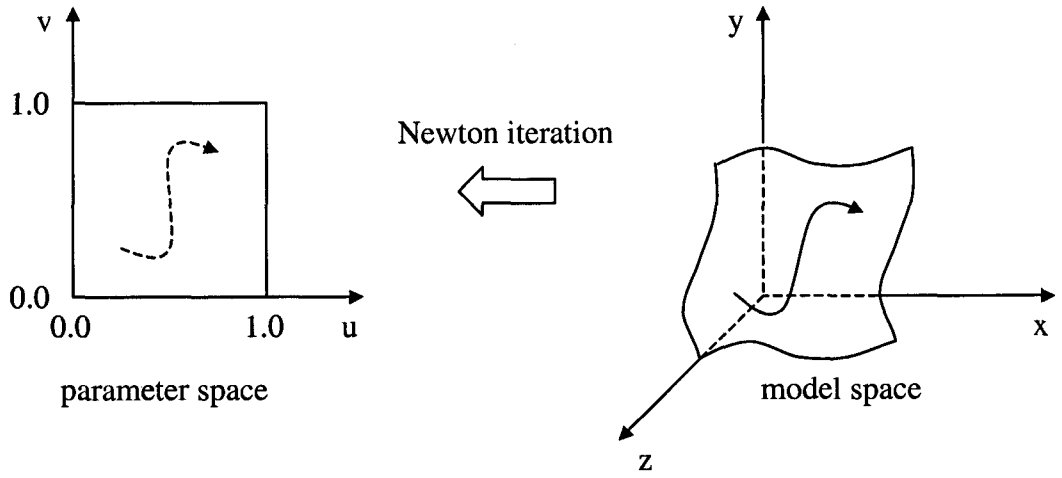


Figure A.6 Inverse mapping from model space to parameter space.

A.7. NURBS Curve Splitting

Curve splitting, or subdivision, is the act of breaking apart one continuous curve into several discontinuous curves. Inserting a knot value $\bar{u}$ into the knot vector until the knot is repeated p, or degree, times splits the curve at that knot value. New control points for the split curves are also created. Visually, the split curves together do not appear any different than what the original curve looked like, as shown in Figure A.7.

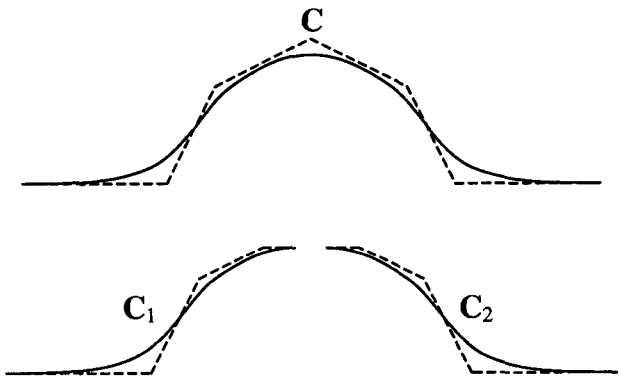


Figure A.7 Splitting a NURBS curve [19].

A.8. NURBS Curve Normalization

Normalizing a NURBS curve confines the parametric range to [0,1]. This is done if the knot vector range is something other than this traditional default. Since the knot vector bounds parameter space, the control points of the parameter space curves also need to be normalized. For example, if the following knot vectors for a cubic surface are defined

$$U = \left\{-1,-1,-1,-1,-\frac{1}{2},0,\frac{1}{2},1,1,1,1\right\}$$

$$V = \left\{0,0,0,0,\frac{1}{2},\frac{1}{2},\frac{3}{4},1,1,1,1\right\}$$

then the control points in the u-direction are bound by [-1,1] and in the v-direction by [0,1]. For this example, the v-direction is already normalized. The U knot vector when normalized would look just like the V knot vector does now.

Appendix B. Types of Geometric Models

The use of CAD in industry for design, engineering, and manufacturing has proven to be beneficial in shortening the development cycle, thus lowering cost, as well as improving the reliability and quality of a product. CAD systems, therefore, must represent geometrical model information in an elegant manner to the designer. To that end, conventional CAD systems have three basic ways of representing 3D geometric models [26] [69]:

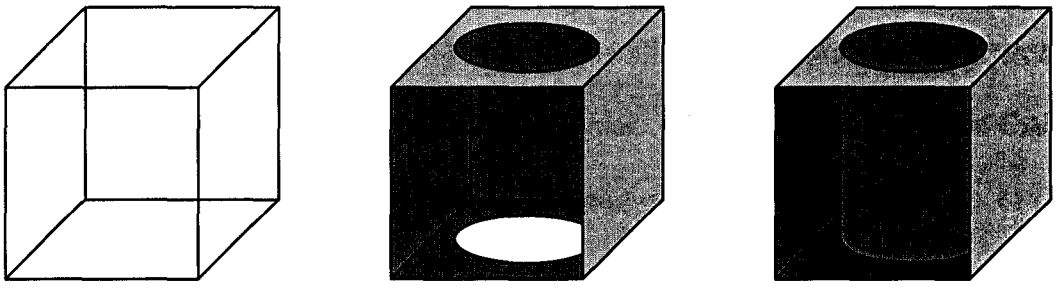
- Wireframe models
- Surface models
- Solid models

Wireframe models support the generation of typically 2D engineering drawings using points, lines, arcs and circles, conics, and curves. The curve representations may be Hermite splines, Bézier curves, B-spline curves, and NURBS curves. Although all modern CAD systems support wireframe representation, it is not a very useful visual format for complex objects.

Surface models support the design and manufacture of complex sculptured surfaces. The surface representations may be planar surfaces, ruled surfaces, surfaces of revolution, tabulated cylinders, Hermite surfaces, Bézier surfaces, B-spline surfaces, Coons surfaces, blending surfaces, offset surfaces, triangular patches, sculptured surfaces, and NURBS surfaces. Surface models provide accurate visual feedback, but are lacking for analysis purposes.

Solid models completely define 3D geometry for higher level analysis and computing. There are several methods of solid model representation that are covered in the next section.

Figure B.1 shows a bored cube block as a wireframe, surface, and solid model with the front face removed from the surface and solid models for visual purposes. The advantages and disadvantages of each geometric type are given in Table B.1, Table B.2, and Table B.3.



(a) Wireframe Model (b) Surface Model (c) Solid Model

Figure B.1 Wireframe, surface, and solid models of a bored cube block.

Table B.1 Advantages and disadvantages of wireframe models.

Advantages	Disadvantages
Simply and uniquely constructed	Verbose
Straightforward to use	Creates nonsense objects
Forms the basis for surface models	Visually ambiguous
Most computationally economical to use	Lack of visual coherence
Dimensional information	No volume or mass information

Table B.2 Advantages and disadvantages of surface models.

Advantages	Disadvantages
More complete model than wireframe	No topology information
Less ambiguous than wireframe	No mass information
Uniquely constructed	More difficult to use than wireframes
Provides dimensional and volume information	More complex to construct than wireframes

Table B.3 Advantages and disadvantages of solid models.

Advantages	Disadvantages
Most complete model	Not uniquely constructed
Provides topology information	
Easiest definition of a model	
Unambiguous	
Provides dimensional, volume, and mass information	

There are several methods of solid model representation [7] [26] [69]. These include sweep representations, (pure) primitive instancing, and three categories of representations based on their mode of creation: decomposition models, constructive models, and boundary models. Within these classes are various subclasses of creation types: decomposition models are specified by cell decompositions, spatial occupancy or exhaustive enumerations, and adaptive or space or Octree subdivisions; constructive models are specified by half-spaces and CSG; and boundary models are specified by B-Reps. Of these solid model representations, CSG and B-Reps have been the most successful due to their vast modeling domain and support of wide ranges of applications. CSG achieved early prominence in CAD, but B-Reps dominate today's applications.

B.1. CSG

CSG [22] models are represented as trees where the leaves are primitive geometric shapes (cubes, pyramids, cylinders, cones, and spheres) and the interior nodes are either Boolean operations (union, difference and intersection) or transformations (translation, rotation, and scale). The primitives are all bounded since working with bounded objects is

easier than with unbounded half-spaces. The transformations are used to define the positions, orientations and other features of the primitives, while the Boolean operations are used to combine the primitives into more complex objects. CSG models are unambiguous, compact because of a simple data structure, concise and valid if all primitives are valid, but they are not unique and are limited in the types of solids that can be produced due to the finite number of primitives defined [70] [71]. Figure B.2 illustrates a simple CSG model.

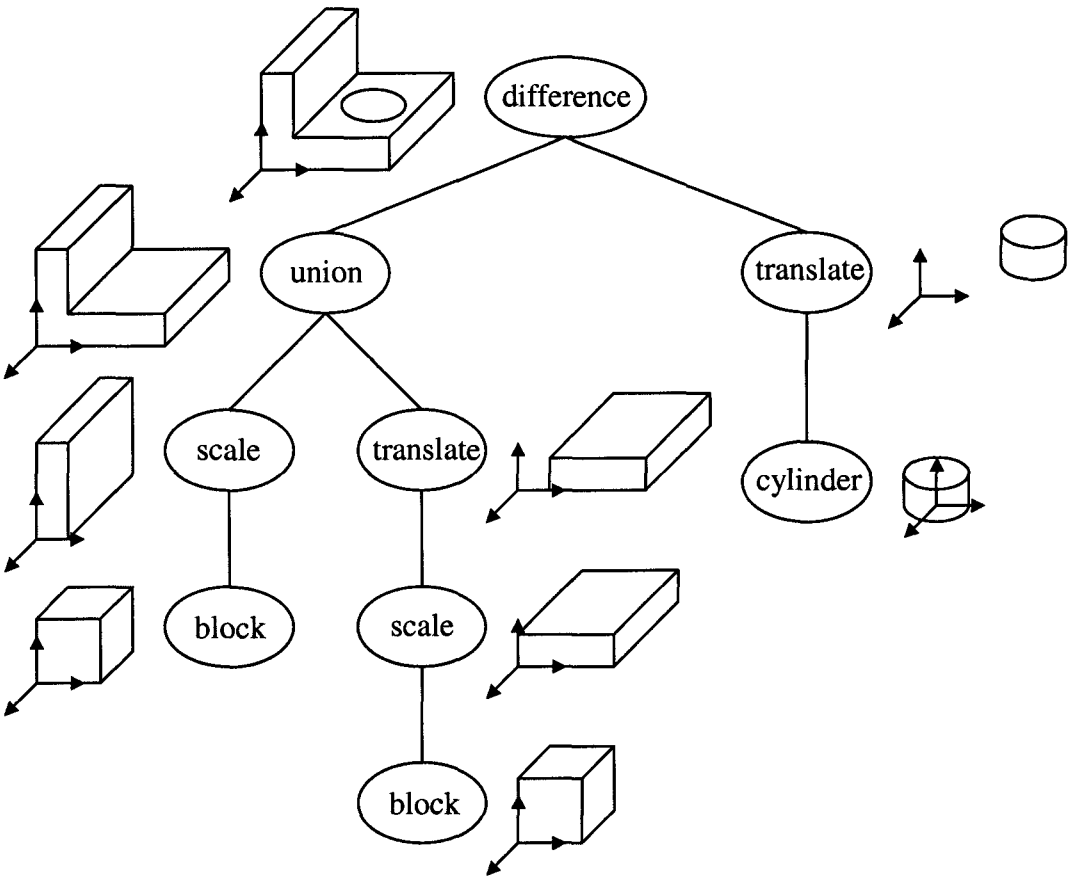


Figure B.2 A simple CSG model.

B.2. B-Reps

Boundary models [72] provide a full and complete representation of a solid in terms of its boundaries. The B-Rep is the standard description method for boundary models. They define solids in terms of their boundary and the topological information that defines the relationships between the vertices, edges, and faces that make up the boundary. The topological information allows a solid to be represented as a closed space, thus, allowing the boundary to define interior, exterior and surface points of the solid. The faces are normally bounded regions of planar, quadratic, toroidal or sculptured surfaces. Closed clockwise directional curves lying on the surfaces represent the bounded regions of the surfaces that form the faces. Faces may have several closed counter-clockwise bounding curves to represent holes in the solid as long as each such curve has a clockwise directional curve encompassing it. These closed curves are normally referred to as trim curves.

Valid B-Rep models satisfy the following conditions to disallow self-intersecting and open objects [70]:

- The set of faces forms a complete skin of the solid with no missing parts
- Faces do not intersect each other except at common vertices or edges
- Boundaries of faces do not intersect themselves
- Each edge must connect two vertices
- Each edge must be shared by exactly two faces
- At least three edges must meet at each vertex

B-Reps were originally developed with a user interface based on the construction of valid faces via Euler operators. Later CAD development replaced this user interface with more intuitive alternatives, but the B-Rep remains the most popular method for persisting solid models in memory.

In the following example, Figure B.3 shows a rectilinear block with the vertices, edges, and faces labeled, Figure B.4 shows the general winged-edge structure, and Table B.4 is the winged-edge data structure for the edge-based boundary representation where e_{id} is the edge, v_{start} and v_{end} are the start and end vertices of the edge, f_{cw} and f_{ccw} are the clockwise and counter-clockwise directional face the edge belongs to, n_{cw} and p_{cw} are the next and previous edges to the edge in the clockwise direction, and n_{ccw} and p_{ccw} are the next and previous edges to the edge in the counter-clockwise direction.

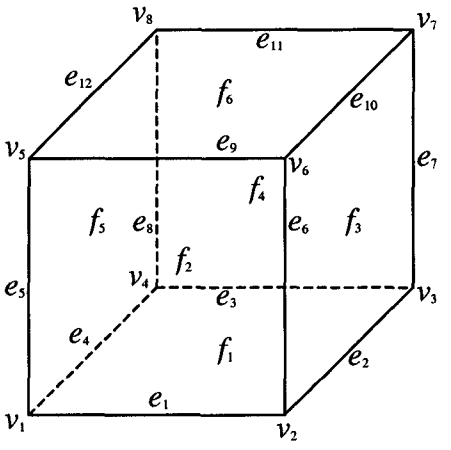


Figure B.3 Rectilinear block [73].

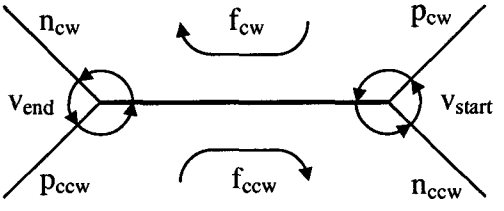


Figure B.4 Winged-edge [73].

Table B.4 Winged-edge data structure [73].

e_{id}	v_{start}	v_{end}	f_{cw}	f_{ccw}	n_{cw}	p_{cw}	n_{ccw}	p_{ccw}
e_1	v_1	v_2	f_1	f_2	e_2	e_4	e_5	e_6
e_2	v_2	v_3	f_1	f_3	e_3	e_1	e_6	e_7
e_3	v_3	v_4	f_1	f_4	e_4	e_2	e_7	e_8
e_4	v_4	v_1	f_1	f_5	e_1	e_3	e_8	e_5
e_5	v_1	v_5	f_2	f_5	e_9	e_1	e_4	e_{12}
e_6	v_2	v_6	f_3	f_2	e_{10}	e_2	e_1	e_9
e_7	v_3	v_7	f_4	f_3	e_{11}	e_3	e_2	e_{10}
e_8	v_4	v_8	f_5	f_4	e_{12}	e_4	e_3	e_{11}
e_9	v_5	v_6	f_2	f_6	e_6	e_5	e_{12}	e_{10}
e_{10}	v_6	v_7	f_3	f_6	e_7	e_6	e_9	e_{11}
e_{11}	v_7	v_8	f_4	f_6	e_8	e_7	e_{10}	e_{12}
e_{12}	v_8	v_5	f_5	f_6	e_5	e_8	e_{11}	e_9

Appendix C. Implicit vs. Parametric Equations

Implicit equations and parametric functions are the two most common ways of representing curves and surfaces for geometric modeling. Implicit equations define an implicit relationship between variables, such as $f(x,y)=0$ for curves, or $f(x,y,z)=0$ for surfaces. For example, a circle of unit radius centered at the origin, Figure C.1, is specified by the equation

$$f(x,y)=x^2+y^2-1=0$$

For the parametric form, a single independent parameter u is explicitly mapped to each coordinate of a point on the curve, such as $\mathbf{C}(u)=(x(u),y(u))$ for curves, or $\mathbf{S}(u,v)=(x(u,v),y(u,v),z(u,v))$ for surfaces with $a\le u\le b$ where the interval $[a,b]$ is arbitrary but usually normalized to [0,1]. For example, the same circle of unit radius centered at the origin, Figure C.1, is defined by the parametric functions

$$x(u)=\cos(u)$$

$$y(u)=\sin(u) \qquad 0\le u\le 2\pi$$

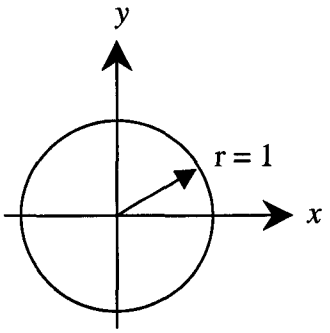


Figure C.1 Unit circle centered at the origin [19].

With the implicit form, it is simple to determine if a given point (x, y) or (x, y, z) lies on the curve or surface. However, it is difficult to calculate points on the curve or surface unless the equation can be reduced to an explicit equation for x any y or for x, y, and z. On the other hand, for the parametric form, points on the curve or surface are easily computed but it is difficult to determine if a given point lies on the curve or surface [20].

The use of parametric equations has gained significant prominence within modern CAD systems, although there are several implicit/parametric hybrid systems. Parametric equations are popular because they offer more degrees of freedom for controlling the shape of curves and surfaces, perform transformations easily and directly, handle infinite slopes, separate the roles of the dependent and independent variable, and are inherently bounded [8].

Appendix D. Parametric vs. Variational Modeling

Parametric modeling, also known as geometric constraint modeling, is the term known for the use of constraints to determine the spatial relationships of various features of CAD models. This allows a designer the flexibility to modify the topology and dimensions of a model without having to entirely recreate the model from scratch. There exist several methods of parametric modeling, but only two are of fundamental interest: the parametric and variational approaches [26] [74].

The design processes for both the parametric and variational approach are similar [7], as shown in Figure D.1. The first step is to define the topology of the model. This results in a model with the desired geometric elements and connectivity between those elements. The second step is to define the constraints between the geometric elements. These constraints specify the mathematical relationship between the numerical variables of the geometric elements. The types of constraints defined are ground, dimensional, geometric, and algebraic constraints, as shown in Figure D.2. The third step is to evaluate the model. This results in a model with the desired topology satisfying the defined constraints. However, not all systems of constraints can be solved. The fourth step, if desired, is to modify the constraint values to create a variant of the model.

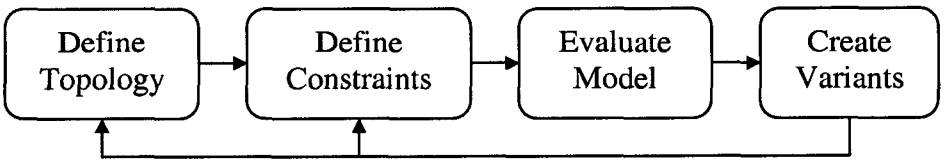


Figure D.1 Parametric and variational design process.

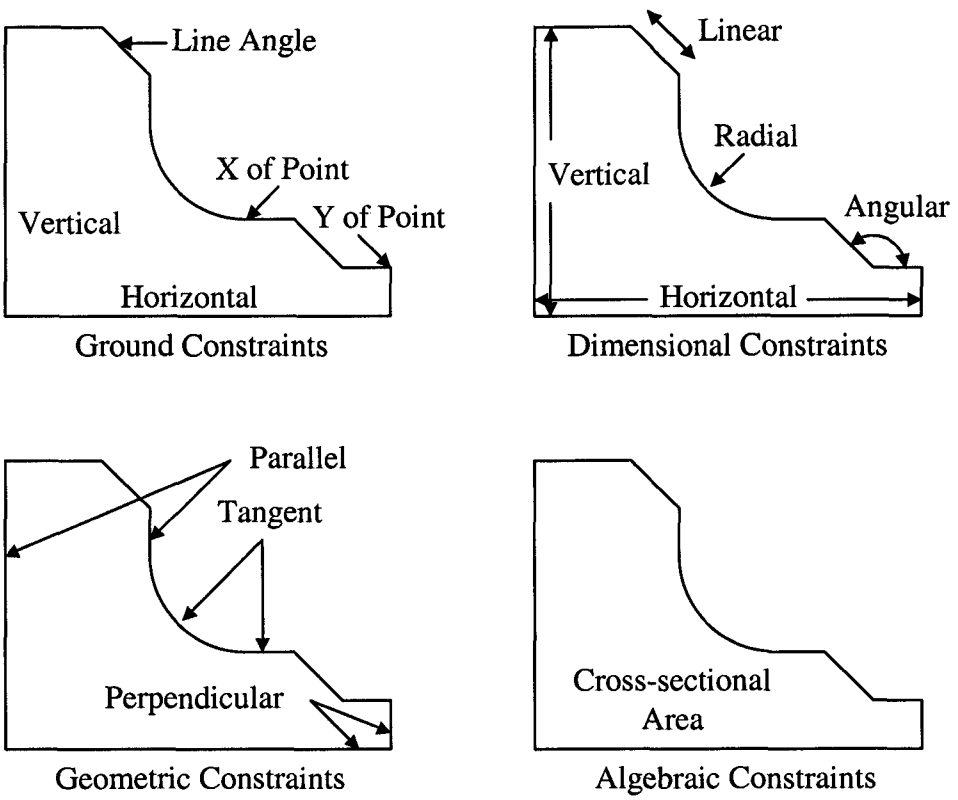


Figure D.2 Types of constraints [26].

One difference between the two approaches is the method used to evaluate the model in the third step [74] [75], as shown in Figure D.3. The parametric approach makes use of sequential solvers on a governing set of uncoupled equations. The variational approach uses simultaneous solvers on equations that may be either coupled or uncoupled. Another difference is in how the model is constrained in the second step [74]. The parametric approach requires that the model be fully constrained, where as the variational approach can be fully or partially constrained. For the parametric approach, a designer needs to anticipate the types of changes to the constraints so the design can be solved sequentially. The variational approach allows a designer more flexibility to experiment with the model even if

it is incomplete. Table D.1 and Table D.2 list the advantages and disadvantages of the parametric and variational approaches [75].

$$\begin{bmatrix} A_{11} & 0 & 0 & 0 \\ 0 & A_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & A_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Parametric Approach Variational Approach

Figure D.3 Parametric vs. variational matrix approaches [76].

Table D.1 Advantages and disadvantages of the parametric approach.

Advantages	Disadvantages
Models many parts	Cannot model some parts
Easy for simple situations	Not appropriate for many parts
Fastest response time	Cannot handle coupled equations
Geometry does not flip	Must be fully constrained at all times
Handles uncoupled geometry and equations	

Table D.2 Advantages and disadvantages of the variational approach.

Advantages	Disadvantages
Easy to construct models for even complex design situations	Somewhat less performance due to decomposition of equation set
Allows under-constrained models for early design iterations	Can allow geometry to flip into different configuration
Handles coupled geometry and equations	
Models all kinds of parts	
Provides sensitivity information for tolerance analysis, mechanisms analysis and design optimization	

Appendix E. Topological Challenges in Surface Modeling

Many challenges in representing and defining models are caused by how mathematical topology is addressed in surface modeling. Although surface modeling attempts to maintain strict topology, this is not always feasible since surface modeling requires a consistent mathematical representation that can be solved computationally. These differences are noticeable when comparing the topological space and parametric space of objects. For surfaces of objects that can be defined, or nearly defined, rectilinearly, this does not pose a problem, which is the majority of cases. In these cases, the topological and parametric space definitions are identical, as seen in Figure E.1(a) and (c). The representative planar 3D objects are shown in Figure E.1(b) and (d). The topological space follows the typical convention as developed by mathematics. The parametric space follows the trim curve counter-clockwise bounded rule as defined by surface modeling.

Problems occur when the topological space representation begins to fold on itself, as with a sphere, cylinder, or cone causing a pole, seam, or degenerate edge problem in surface modeling. The problem comes not from the definition of the object, but from the programmatic implementation of their definition within CAD packages. The topological and parametric space definitions for a sphere, cone, and cylinder and their 3D representation are shown in Figure E.2, Figure E.3, and Figure E.4 respectively.

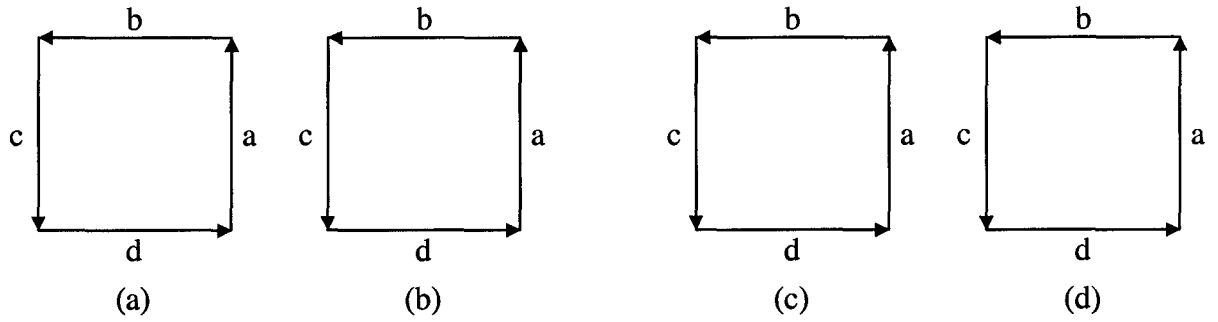


Figure E.1 Topological and parametric space of a plane.

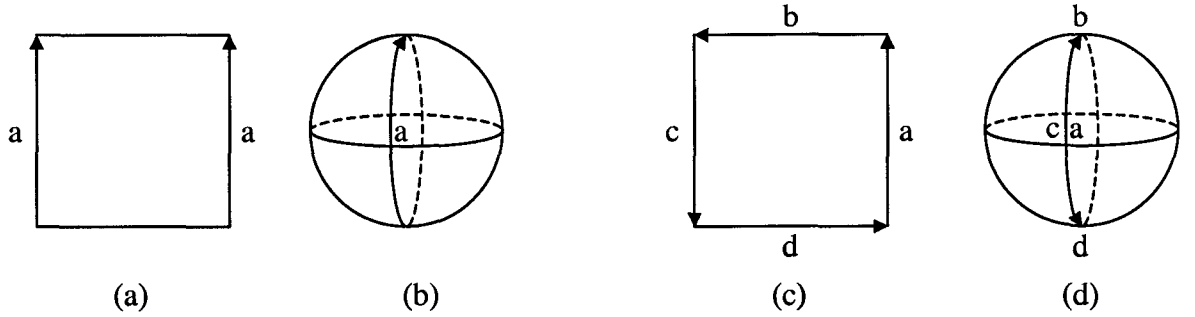


Figure E.2 Topological and parametric space of a sphere (pole).

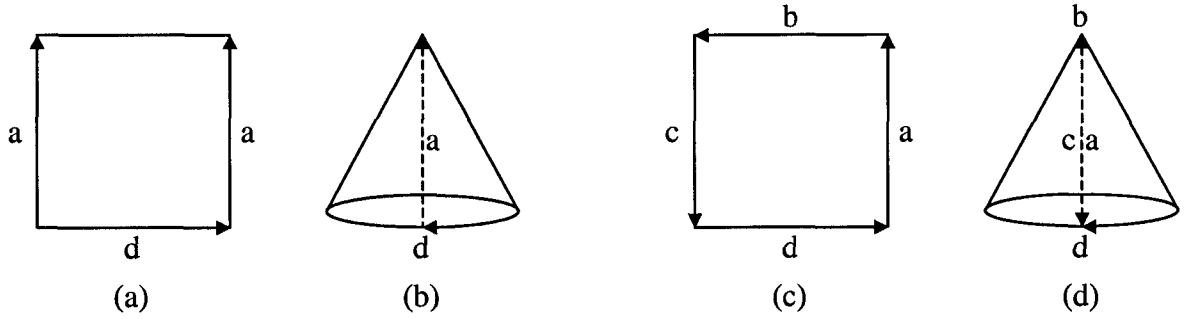


Figure E.3 Topological and parametric space of a cone (degenerate edge).

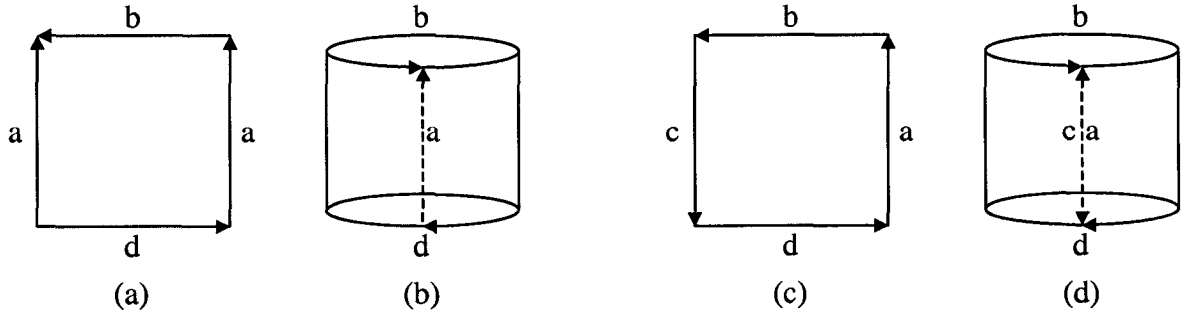


Figure E.4 Topological and parametric space of a cylinder (seam).

For the sphere, topology only defines one edge that abuts on itself. The other pseudo edges are the poles and they vanish to nothing. For surface modeling, keeping with the consistent definition theme of surfaces, the topological representation is invalid. Therefore, a parametric definition similar to the rectilinear case is needed. How this is implemented programmatically is a cause of many headaches and problems. The same is true for the cone that defines two edges and for the cylinder that defines three edges. The full complement of parametric space curves is shown in each of the figures, but that does not necessitate that each CAD vendor defines these problem objects this way. They may special case these definitions in a manner so they are handled by their CAD system, but this does not necessitate the free flow of geometric data between different systems.

Appendix F. Sample IGES File

An IGES file is composed of fixed eighty column records divided into either five or six sections of information: Flag, Start, Global, Directory Entry, Parameter Data, and Terminate. The Flag section, denoted by an F in column 73, is only present when the file is either in binary or compressed ASCII form. The Start section, denoted by an S in column 73, is a header providing a human-readable prologue to the file. The Global section, denoted by a G in column 73, contains preprocessor and postprocessor information needed to handle the file. The Directory Entry section, denoted by a D in column 73, provides an index for the file and contains attribute information for each entity. The Parameter Data section, denoted by a P in column 73, contains the specific entity definition. The Terminate section, denoted by a T in column 73, represents the end of the file and must be the last line in the file. A file may contain any number of different entity types necessary to represent a model. Each entity consists of a directory entry and parameter data. Bi-directional pointers exist between the directory entry and the parameter data for each entity.

The following is a sample IGES file of a rational b-spline curve entity (type 126) [77]. For a thorough description of the IGES standard and this file structure, consult the latest IGES specification [78].

```
1H,,1H;,5HF126X,9HF126X.IGS,13H{unspecified},13H{unspecified},32,38,15, G      1
308,15,5HF126X,1.,1,2HIN,4,0.01333,13H950612.040122,0.0001,178.,        G      2
31HIGES Version 5.3 Edit Committee,9HIPO/USPRO,11,0,13H940219.133140,;  G      3
     126       1       0       1       2       0       0       000000001D      1
     126       3       5       3       0                               D      2
     116       4       0       1       0       0       0       000000001D      3
     116       2       6       1       0                               D      4
     116       5       0       1       0       0       0       000000001D      5
     116       2       6       1       0                               D      6
     116       6       0       1       0       0       0       000000001D      7
     116       2       6       1       0                               D      8
     116       7       0       1       0       0       0       000000001D      9
     116       2       6       1       0                               D     10
```

```
     116       8       0       1       0       0       0       000000001D     11
     116       2       6       1       0                               D      12
     116       9       0       1       0       0       0       000000001D     13
     116       2       6       1       0                               D      14
     106      10       0       5       0       0       0       000000001D     15
     106       2       7       2      12                               D      16
126,5,3,1,0,1,0,0.,0.,0.,0.,0.333333,0.666667,1.,1.,1.,1.,1.,1.,       1P      1
1.,1.,1.,1.,-178.,109.,0.,-166.,128.,0.,-144.,109.,0.,-109.,           1P      2
112.,0.,-106.,134.,0.,-119.,138.,0.,0.,1.,0.,0.,1.;                    1P      3
116,-178.,109.,0.,0;                                                   3P      4
116,-166.,128.,0.,0;                                                   5P      5
116,-144.,109.,0.,0;                                                   7P      6
116,-109.,112.,0.,0;                                                   9P      7
116,-106.,134.,0.,0;                                                  11P      8
116,-119.,138.,0.,0;                                                  13P      9
106,2,6,-178.,109.,0.,-166.,128.,0.,-144.,109.,0.,-109.,112.,0.,      15P     10
-106.,134.,0.,-119.,138.,0.;                                          15P     11
S      1G      3D     16P     11                                       T       1
```

Appendix G. Sample STEP File

STEP is organized as a series of six separately published classes: description methods, implementation methods, conformance methodology, common resources, APs, and abstract test suites. The description methods class, parts 1-19, provides an overview of STEP as well as the description for the unambiguous data-modeling language EXPRESS. The implementation class, parts 20-29, describes the mapping from the formal specification to the implementation representation. The conformance methodology class, parts 30-39, provides information on software-product conformance testing, abstract-test suite creation, and testing laboratory responsibilities. The common resources class, formerly called integrated-information resources class, contains the generic STEP data models and is considered the building blocks of STEP. There are four categories of common resources: integrated-generic resources and integrated-application resources (IR), application-interpreted constructs (AIC), and application modules (AM). The integrated-generic resources, parts 40-59, are generic entities that are used by APs as needed. The integrated-application resources, 100 series parts, are entities that contain slightly more context than the integrated generic resources. The application-interpreted constructs, 500 series parts, are reusable groups of information-resource entities that make it easier to express identical semantics in more than one AP. The application modules, 1000 series parts, are reusable groups of functional information requirements of applications that extend the AIC capability. The APs class, 200 series parts, are industry specific data models that describe what data is to be used in describing a product and how the data is to be used in the model. The abstract test suites class, 300 series parts,

consists of test data and criteria used for assessing the conformance of STEP software products to the associated AP.

The EXPRESS modeling language is used to construct the physical file. For example, a point and circle are defined by:

```
ENTITY point;
     x: REAL;
     y: REAL;
     z: OPTIONAL_REAL;
END_ENTITY;

ENTITY circle;
     center: point;
     radius: REAL;
END_ENTITY;
```

A circle of unit radius centered at the origin would use an instance of the point and circle entities and written to the physical file as:

```
#10=POINT(0.0,0.0,$);
#20=CIRCLE(#10,1.0);
```

The following is a sample STEP file of a rational b-spline curve (AP203). For a thorough description of the STEP standard and this file structure, consult the latest STEP specification from the ISO [79].

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('','2;1');
FILE_NAME('F126X','2003-02-17T23:34:54',('unknown'),(''),
'PRO/ENGINEER BY PARAMETRIC TECHNOLOGY CORPORATION, 2001150',
'PRO/ENGINEER BY PARAMETRIC TECHNOLOGY CORPORATION, 2001150','');
FILE_SCHEMA(('CONFIG_CONTROL_DESIGN'));
ENDSEC;
DATA;
#1=DIMENSIONAL_EXPONENTS(1.E0,0.E0,0.E0,0.E0,0.E0,0.E0,0.E0);
#2=(LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
#3=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.54E1),#2);
#4=(CONVERSION_BASED_UNIT('INCH',#3)LENGTH_UNIT()NAMED_UNIT(#1));
#5=DIMENSIONAL_EXPONENTS(0.E0,0.E0,0.E0,0.E0,0.E0,0.E0,0.E0);
#6=(NAMED_UNIT(*)PLANE_ANGLE_UNIT()SI_UNIT($,.RADIAN.));
#7=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(1.745329251994E-2),#6);
#8=(CONVERSION_BASED_UNIT('DEGREE',#7)NAMED_UNIT(#5)PLANE_ANGLE_UNIT());
#9=(NAMED_UNIT(*)SI_UNIT($,.STERADIAN.)SOLID_ANGLE_UNIT());
```

```
#10=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(9.259894164471E-3),#4,
'closure',
'Maximum model space distance between geometric entities at asserted
connectivities');
#11=(GEOMETRIC_REPRESENTATION_CONTEXT(3)GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT(
(#10))GLOBAL_UNIT_ASSIGNED_CONTEXT((#4,#8,#9))REPRESENTATION_CONTEXT('ID1',
'3'));
#12=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#13=DIRECTION('',(0.E0,0.E0,1.E0));
#14=DIRECTION('',(1.E0,0.E0,0.E0));
#17=APPLICATION_CONTEXT(
'CONFIGURATION CONTROLLED 3D DESIGNS OF MECHANICAL PARTS AND ASSEMBLIES');
#18=APPLICATION_PROTOCOL_DEFINITION('international standard',
'config_control_design',1994,#17);
#19=DESIGN_CONTEXT('',#17,'design');
#20=MECHANICAL_CONTEXT('',#17,'mechanical');
#21=PRODUCT('F126X','F126X','NOT SPECIFIED',(#20));
#22=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('','LAST_VERSION',#21,
.MADE.);
#26=PRODUCT_CATEGORY('part','');
#27=PRODUCT_RELATED_PRODUCT_CATEGORY('detail','',(#21));
#28=PRODUCT_CATEGORY_RELATIONSHIP('','',#26,#27);
#29=SECURITY_CLASSIFICATION_LEVEL('unclassified');
#30=SECURITY_CLASSIFICATION('','',#29);
#31=CC_DESIGN_SECURITY_CLASSIFICATION(#30,(#22));
#32=APPROVAL_STATUS('approved');
#33=APPROVAL(#32,'');
#34=CC_DESIGN_APPROVAL(#33,(#30,#22,#23));
#35=CALENDAR_DATE(103,17,2);
#36=COORDINATED_UNIVERSAL_TIME_OFFSET(6,0,.BEHIND.);
#37=LOCAL_TIME(23,34,5.4E1,#36);
#38=DATE_AND_TIME(#35,#37);
#39=APPROVAL_DATE_TIME(#38,#33);
#40=DATE_TIME_ROLE('creation_date');
#41=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#38,#40,(#23));
#42=DATE_TIME_ROLE('classification_date');
#43=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#38,#42,(#30));
#44=PERSON('UNSPECIFIED','UNSPECIFIED',$,$,$,$);
#45=ORGANIZATION('UNSPECIFIED','UNSPECIFIED','UNSPECIFIED');
#46=PERSON_AND_ORGANIZATION(#44,#45);
#47=APPROVAL_ROLE('approver');
#48=APPROVAL_PERSON_ORGANIZATION(#46,#33,#47);
#49=PERSON_AND_ORGANIZATION_ROLE('creator');
#50=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#46,#49,(#22,#23));
#51=PERSON_AND_ORGANIZATION_ROLE('design_supplier');
#52=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#46,#51,(#22));
#53=PERSON_AND_ORGANIZATION_ROLE('classification_officer');
#54=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#46,#53,(#30));
#55=PERSON_AND_ORGANIZATION_ROLE('design_owner');
#56=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#46,#55,(#21));
#15=AXIS2_PLACEMENT_3D('',#12,#13,#14);
#16=SHAPE_REPRESENTATION('',(#15),#11);
#23=PRODUCT_DEFINITION('design','',#22,#19);
#24=PRODUCT_DEFINITION_SHAPE('','SHAPE FOR F126X.',#23);
#25=SHAPE_DEFINITION_REPRESENTATION(#24,#16);
ENDSEC;
END-ISO-10303-21;
```

Bibliography

[1] Jan Helge Bohn. *Automatic Cad-Model Repair.* Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, New York, August 1993.

[2] Rida T. Farouki. Closing the Gap Between CAD Model and Downstream Application. *SIAM News*, 32(5), June 1999, Retrieved August 15, 2002 from http://www.siam.org/siamnews/06-99/cadmodel.htm.

[3] Engine Tuning – Data Translation at Cosworth. *CADserver*, October 30, 2001, Retrieved September 26, 2002 from http://www.cadserver.co.uk/common/viewer/archive/2001/Oct/30/feature5.phtm.

[4] Intelligent Data Translation: How Close Are We? *CADserver*, November 1, 2000, Retrieved September 26, 2002 from http://www.cadserver.co.uk/common/viewer/archive/2000/Nov/1/feature4.phtm.

[5] S. Utterdyke. Curing IGESphobia: Translating and Repairing Problem CAD Models. *ProE: The Magazine*, November 2001.

[6] Smita B. Brunnermeier and Sheila A. Martin. *Interoperability Cost Analysis of the U.S. Automotive Supply Chain.* 99-1 Planning Report, Research Triangle Institute, March 1999.

[7] Jami J. Shah and Martti Mäntylä. *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications.* John Wiley & Sons, New York, 1995.

[8] Michael E. Mortenson. *Geometric Modeling.* John Wiley & Sons, New York, 1985.

[9] Pierre E. Bezier. The First Years of CAD/CAM and the UNISURF CAD System. In Les Piegl (Ed.), *Fundamental Developments of Computer-Aided Geometric Modeling*, Academic Press, San Diego, California, 1993.

[10] Pierre E. Bezier. *Numerical Control: Mathematics and Applications*. John Wiley & Sons, New York, 1972.

[11] Steven A. Coons. *Surfaces for Computer-Aided Design of Space Forms*. Technical Report MAC-TR-41, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1967.

[12] Paul de Casteljau. *Formes a poles*. Note Citroën, 1959.

[13] James C. Ferguson. Multi-variable Curve Interpolation. *Journal of the ACM*, 11(2):329-346, 1963.

[14] William J. Gordon. Blending Function Methods of Bivariate and Multi-variate Interpolation and Approximation. *SIAM Journal of Numerical Analysis*, 8(1):158-177, 1971.

[15] Maurice G. Cox. *Numerical Methods for the Interpolation and Approximation of Data by Spline Functions*. Ph.D. Thesis, City University, London, 1975.

[16] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[17] Richard F. Riesenfeld. *Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design*. Ph.D. Thesis, Syracuse University, New York, 1968.

[18] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bezier and B-Spline Techniques*. Springer-Verlag, Berlin, 2002.

[19] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, Germany, 1997.

[20] Ivor D. Faux and Michael J. Pratt. *Computational Geometry for Design and Manfacture*. John Wiley & Sons, New York, 1979.

[21] Ivan Edward Sutherland. *SKETCHPAD: A Man-Machine Graphical Communication System*. Ph.D. Thesis, Massachusetts Institute of Technolgy, Cambridge, Massachusetts, 1963.

[22] Aristides A.G. Requicha and Herbert B. Voelcker. *Constructive Solid Geometry*. Technical Memo No. 15, Production Automation Project, University of Rochester, Rochester, New York, November 1977.

[23] I.C. Braid. *Notes on a Geometric Modeler*. Computer Laboratory, CAD Group Document 101, University of Cambridge, Cambridge, England, 1979.

[24] A. A. G. Requicha. *Mathematical Models of Rigid Solid Objects*. Technical Memo No. 29, Production Automation Project, University of Rochester, New York, 1978.

[25] H. B. Voelcker and A. A. G. Requicha. *Boundary Evaluation Procedures for Objects Defined via Constructive Solid Geometry*. Technical Memo No. 26, Production Automation Project, University of Rochester, New York, 1980.

[26] Chris McMahon and Jimmie Browne. *CAD/CAM: Principles, Practice and Manufacturing Management*. Addison-Wesley, Harlow, England, 1998.

[27] Peter R. Wilson. A Short History of CAD Data Transfer Standards. *IEEE Computer Graphics and Applications*, 7(6):64-67, June 1987.

[28] Ibrahim Zeid. *CAD/CAM Theory and Practice*. McGraw-Hill, New York, 1991.

[29] *NF Z68-300: Industrial Automation, Data Exchange and Transfer Standard Specification*. AFNOR, December 1993, Purchase at http://www.boutique.afnor.fr/.

[30] *DIN 66301: Industrial Automation; Computer-Aided Design; Format for the Exchange of Geometrical Informations.* Beuth Verlag GmbH, Berlin, Germany, 1986, Purchase at http://www2.din.de/.

[31] B. L. M. Goldstein, S. J. Kemmerer, and C. H. Parks. *A Brief History of Early Product Data Exchange Standards.* National Institute of Standards and Technology Information/Internal Report 6221 (NISTIR 6221), September 2, 1998.

[32] *CAD*I, CAD Interfaces.* European Strategic Program for Research in Information Technology, Retrieved September 17, 2003 from http://www.cordis.lu/esprit/.

[33] *VDMA/VDA 66319.* Beuth Verlag GmbH, Berlin, Germany, 1989, Purchase from http://www2.din.de/.

[34] *Contribution of the United Kingdom, AECMA: Report of Geometry Data Exchange Study Group.* SC4 N004, June 6, 1984, Request from http://www.tc184-sc4.org/-SC4_Open/SC4_and_Working_Groups/SC4_N-DOCS/1-499/maindisp.cfm.

[35] *CAM-I Geometric Modeling Project Boundary File Design (XBF-2).* CAM-I Report R-81-GM-02.1, CAM-I, Arlington, Texas, April 1981.

[36] *Experimental Solids Proposal (ESP).* IGES Internal Report, NBS, Gaithersburg, Maryland, September 1984.

[37] *Electronic Design Interchange Format.* EDIF Technical Centre, Retrieved on October 20, 2002 from http://www.edif.org/.

[38] *DIN V 40940: Neutral interchange format for diagram data (VNS); format for the interchange of the documentation of electrotechnical plants.* Beuth Verlag GmbH, Berlin, Germany, 1992, Purchase from http://www.din2.de/.

[39] *A Brief History of VHDL.* Doulos Limited, Retrieved October 11, 2002 from http://-www.doulos.com/knowhow/vhdl_designers_guide/a_brief_history_of_vhdl.

[40] International Electrotechnical Commission: Technical Committee TC3: Information Structures, Documentation, and Graphical Symbols, Retrieved on October 14, 2002 from http://tc3.iec.ch/.

[41] *IPC-D-350, Printed Board Description in Digital Form.* IPC, Purchase at http://www.ipc.org/.

[42] Electronic Industries Alliance, Visited December 12, 2002 to http://www.eia.org/.

[43] CAD – From NEDO to Round Table. *INTERCHANGE*, Issue 7, 1995, Retrieved on November 16, 2002 from http://www.jiscmail.ac.uk/files/EDI-PROCESS-INDUSTRIES/nlet7.txt.

[44] The Business Roundtable, Visited November 17, 2002 to http://www.business-round-table.co.uk/.

[45] Stephen Wolfe. Fixing Bad CAD Data. *Computer Aided Design Report*, 17(1):4-7, January 1997.

[46] R.J. Goult and P.A. Sherar (Eds.), *Improving the Performance of Neutral File Data Transfers (Research Reports ESPRIT, Project 322, CAD Interfaces)*, Vol. 6. Springer-Verlag, New York, 1987.

[47] F.-L. Krause, C. Stiel and J. Lüddemann. Processing of CAD-Data – Conversion, Verification and Repair. *Proceedings of the 4th ACM Symposium on Solid Modeling and Applications*, pp. 248-254, Atlanta, Georgia, May 14-16, 1997.

[48] N. Anders Petersson & Kyle K. Chand. Detecting Translation Errors in CAD Surfaces and Preparing Geometries for Mesh Generation. *Proceedings of the 10th International Meshing Roundtable*, pp. 363-371, Newport Beach, California, October 7-10, 2001.

[49] John P. Steinbrenner, Nicholas J. Wyman and John R. Chawner. Fast Surface Meshing on Imperfect CAD Models. *Proceedings of the 9th International Meshing Roundtable*, pp. 33-41, New Orleans, Louisiana, October 2-5, 2000.

[50] Antonio E. Uva, Giuseppe Monno, and Bernd Hamann. A New Method for the Repair of CAD Data with Discontinuities. In F. Caputo, A. Lanzotti and X. Leiceaga (Eds.), *Proceedings of Atti del II Seminario Italo-Espanol - Progettazione e Fattibilita dei Prodotti Industriali (Diseno y Fabricabilidad de los Productos Industriales),* ISBN 88-900081-3-X, pp. 59-75, June 1998.

[51] Alla Sheffer, Ted Blacker and Michel Bercovier. CAD Data Repair Based on Virtual Topology. *Proceedings of DETC'98, 1998 ASME Design Engineering Technical Conferences*, DETC98/DAC-5566, pp. 1-10, Atlanta, Georgia, September 13-16, 1998.

[52] Alla Sheffer, Ted Blacker, Jan Clements and Michel Bercovier. Virtual Topology Operators for Meshing. *Proceedings of the 6th International Meshing Roundtable*, pp. 49-66, Park City, Utah, October 13-15, 1997.

[53] Gill Barequet and Subodh Kumar. Repairing CAD Models. *Proceedings IEEE Visualization '97*, pp. 363-370, Phoenix, Arizona, October 19-24, 1997.

[54] Gill Barequet and Micha Sharir. Filling Gaps in the Boundary of a Polyhedron. *Computer Aided Geometric Design*, 12(2):207-229, March 1995.

[55] Hod Lipson. *Computer Aided 3D Sketching for Conceptual Design.* Ph.D. Thesis, Technion Isreal Institute of Technology, Haifa, Israel, September 1998.

[56] Moshe Shpitalni and Hod Lipson. Classification of Sketch Strokes and Corner Detection using Conic Sections and Adaptive Clustering. *Transactions of the ASME Journal of Mechanical Design*, 119(2):131-135, March 1997.

[57] William Martin, Elaine Cohen, Russell Fish, Peter Shirley. Practical Ray Tracing of Trimmed NURBS Surfaces. *Journal of Graphics Tools*, 5(1):27-52, September 2000.

[58] PDElib. International TechneGroup Incorporated, Retrieved August 11, 2002 from http://www.transcendata.com/products_pdelib.htm.

[59] Timothy J. Tautges. The Common Geometry Module (CGM): A Generic, Extensible Geometry Interface. *Proceedings of the 9th International Meshing Roundtable*, pp. 337-348, New Orleans, Louisiana, USA, October 2-5, 2000.

[60] Silvio Merazzi, Edgar A. Gerteisen, and Andrey Mezentsev. A Generic CAD-Mesh Interface. *Proceedings of the 9th International Meshing Roundtable*, pp. 361-370, New Orleans, Louisiana, USA, October 2-5, 2000.

[61] Andrey A. Mezentsev and Thomas Woehler. Methods and Algorithms of Automated CAD Repair for Incremental Surface Meshing. *Proceedings of the 8th International Meshing Roundtable*, pp. 299-309, South Lake Tahoe, California, October 10-13, 1999.

[62] Geoffrey Butlin and Clive Stops. CAD Data Repair. *Proceedings of the 5th International Meshing Roundtable*, pp. 7-12, Pittsburgh, Pennsylvania, USA, October 10-11, 1996.

[63] CADfix. International TechneGroup Incorporated, Retrieved January 7, 2003 from http://www.transcendata.com/products_cadfix.htm.

[64] GDX. XOX Corporation, Visited January 7, 2003 to http://www.xox.com/.

[65] DesignQA. Avatech Solutions, Inc., Visited January 7, 2003 to http://www.-avatechsolutions.com/.

[66] GeometryQA. Avatech Solutions, Inc., Visited January 7, 2003 to http://www.-avatechsolutions.com/.

[67] TransMagic. TransMagic, Inc., Visited January 7, 2003 to http://www.transmagic.-com/.

[68] Christoph M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann Publishers, San Mateo, California, 1989.

[69] Ibrahim Zeid. *CAD/CAM Theory and Practice*. McGraw-Hill, New York, 1991.

[70] A.A.G. Requicha. Representations for Rigid Solids: Theory, Methods, and Systems. *Computing Surveys*, 12(4):437-465, December 1980.

[71] Y.T. Lee and A.A.G. Requicha. Algorithms for Computing the Volume and Other Integral Properties of Solids. I. Known Methods and Open Issues. *Communications of the ACM*, 25(9):635-650, September 1982.

[72] Ian C. Braid. Boundary Modeling. In Les Piegl (Ed.), F*undamental Developments of Computer-Aided Geometric Modeling*, Academic Press, San Diego, California, 1993.

[73] Martti Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988.

[74] Glenn A. Kramer. *Solving Geometric Constraint Systems: A Case Study in Kinematics.* The MIT Press, Cambridge, Massachusetts, 1992.

[75] Martin D. Schussel and Jack C.H. Chung. Constraint-Based Design Technologies. In Carl Machover (Ed.), *The CAD/CAM Handbook*, McGraw-Hill, New York, 1996.

[76] Wayne McClelland. Variational versus Parametric...it's all really quite simple. May 26, 1995, Retrieved August 16, 2002 from http://www.wamware.com/tresources/vg/-var_param.htm.

[77] Examples of Rational B-Spline Curve Entity. National Institute of Standards and Technology, Retrieved September 19, 2002 from http://www.nist.gov/iges/specfigures/-f126x.igs.

[78] *ANSI/USPRO/IPO 100-1996, Initial Graphics Exchange Specification (IGES) Version 5.3.* U.S. Product Data Association, Purchase from https://www.uspro.org/.

[79] International Organization for Standardization, Visited on October 1, 2002, http://www.iso.ch/.