1993

NURBS based computer interface for rapid prototyping

Janardhanan K. Varadachari
Iowa State University

NURBS based computer interface for rapid prototyping

by

Janardhanan K. Varadachari

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Mechanical Engineering
Major: Mechanical Engineering

Iowa State University
Ames, Iowa
1993

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I express my sincere thanks to my major professor and advisor Dr. Jerry Lee Hall for his help, support and guidance. Throughout the course of my study, he had extended freedom to pursue my own research interests and ideas, while at the same time always provided helpful hints and invaluable guidance. He has been a great source of inspiration for me and I will forever remember the valuable lessons learned through him.

I gratefully acknowledge Dr. James Oliver's confidence-building-remarks and am indebted to him for critically reading and correcting my thesis. I also thank him for introducing me to some of the software that saved me a lot of time and effort.

I thank Dr. Les Miller for being on my committee and taking his valuable time to read through my thesis.

I acknowledge the R. A. Engel CIM Laboratory, Department of Mechanical Engineering and the Iowa Center for Emerging Manufacturing Technology for providing the necessary financial support without which none of this would have been possible. I thank Bigjim, Pat Bergan and Dave of Iowa Center for helping me in preparing a video presentation of my thesis.

Many thanks to Dr. Srikanth Padmanabhan of Cummins Engine Co. for helping me with valuable suggestions and introducing me to the friends and facilities at ISU.

Special thanks to all my friends who made my stay at ISU a pleasant and memorable one, particularly, to the lunch group – Brian, Bruce, Shawn, Joe and the president of BBC - Cookie, to the PR (Pecan Research) group – Gaylord and Bruce and finally to the Indian Mafia – my roommate Bala, Ananth & Vidya, Babu & Geetha, Ramu & Shakti and Varma & Rumki, for their wonderful company.

I am grateful to my father, mother and brother who encouraged me to pursue higher studies. I owe most of what I have achieved up to now to them.

ABSTRACT

Rapid prototyping is a new technology which transforms computer models of geometric shapes into prototypes in a very short time. The state-of-the-art systems currently available perform many of the rapid prototyping functions required. However, they have inherent limitations in terms of the ability of the triangulated data to represent objects with acceptable precision. In this research, a new technique has been developed to accurately represent and laser path plan the 2D cross-sections utilizing the Non-Uniform Rational B-Splines (NURBS) data of an object instead of the traditional polygonal or triangulated data. Its advantage over the existing method is that it offers one common mathematical form for the precise representation of the standard analytical shapes as well as free-form curves and surfaces and it requires fewer data conversions. This method is implemented in C and interfaced with SDRC's I-DEAS, a solid modeler, to obtain the NURBS information. A graphical interface is also provided using HOOPS and GL on the DEC and SGI platforms respectively. By addressing the issue of improved data representation on the CAD file front-end, this research will significantly impact the output of rapid prototyping with improved tolerance and surface finish capabilities. This robust mathematical technique has great potential for application in future rapid prototyping systems.

CHAPTER 1. INTRODUCTION

Rapid prototyping is one of the fastest growing manufacturing techniques in which the conceptual computer models of geometric shapes are turned into prototypes in a matter of hours without the use of traditional tooling. The three most important stages in the current prototyping processes are (i) triangulation of the geometric computer model (either solid or surface model), (ii) slicing of the model into various cross-sectional layers, and (iii) X-Y cross hatching or 2 dimensional path planning of the individual cross-sectional profiles. The state-of-the-art systems currently available perform many of the rapid prototyping functions required. However, they have inherent limitations in terms of the ability of the Stereolithography (STL) format (a planar faceted representation where the surfaces of the object are represented as a set of intersecting triangles) to represent objects with acceptable precision. In attempting to reduce the inaccuracy/error introduced by this approximate method, efforts are being concentrated on improved data representation on the Computer Aided Design (CAD) file front-end. This research effort is aimed at presenting solutions to this problem by means of "Non-Uniform Rational B-Spline (NURBS) based computer interface" that uses precise curve and surface data of an object instead of the traditional polygonal or triangulated data. It also describes the need for such an interface and outlines a new approach to accurately represent and path plan the

cross-sectional profiles of the objects for rapid prototyping.

Research Goals

Few attempts have been made to provide a computer interface for rapid prototyping. In the case of STL interface, the computer model is triangulated or tessellated in the geometric modeling system before it is sent to the rapid prototyping systems [10] for slicing and path planning. Helisys [6] uses the numerical control (NC) data of an object for slicing and path planning. No attempt seems to have been made to use the precise representation of the object in the geometric modeling system as the core for the sectioning and path planning routines for rapid prototyping. The overall objective of this research therefore is to build a NURBS based computer interface that will generate the X-Y cross-hatch patterns for the cross-sectional profiles of an object automatically, based on the precise representation in the geometric modeling systems. The specific goals of this study are:

1. To develop computer routines for the acquisition and representation of the cross-sectional profiles of an object based on Non-Uniform Rational B-Splines (NURBS).

2. To develop a method for the two dimensional (2D) cross-hatch planning of the cross-sectional profiles for the laser scanner post-processor.

3. To implement a prototype system based on the methods developed and to simulate the system response with a graphical user interface.

Need For The Research

Although several parts of the problem of providing a computer interface for rapid prototyping have been attempted, there has been no attempt to provide an interface which uses the precise information from the solid modeling system for path planning the laser curing process. At the beginning of this research, the need for such a computer interface was identified. During the development of such an interface based on a CAD representation, a more fundamental understanding of the process of 2D path planning was obtained and this understanding prompted a reexamination of the concepts currently in use. It also seemed clear that the increasing need for functioning prototypes produced through rapid prototyping systems would necessitate more refined methods of geometric representation. Overall the present research has emphasized the need for developing path planning systems on the basis of the precise information from the geometric modeling system. The problem to be solved in this thesis is defined as:

- **"Given a complete geometric description of the part to be prototyped, design appropriate techniques to retrieve precise cross-sectional information and automatically produce a cross-hatch pattern suitable for the laser post processor"**

Thesis Organization

The second chapter of the thesis presents a literature review. The review focuses on two areas: a brief description on the evolution of geometric modeling systems and contemporary efforts in modeling systems for providing CAD interfaces to rapid

prototyping systems.

Details of the methods to acquire NURBS data of the cross-sectional profiles of an object from a solid modeling system, methods of representing and path planning (2D) are provided in Chapter 3. Chapter 4 explains the details of software implementation. Examples are provided in Chapter 5 to explain the method and how it works with different computer models. The conclusions and scope for future work are summarized in Chapter 6. Appendices provide additional details of the interface development.

CHAPTER 2. LITERATURE REVIEW

The research accomplished in this thesis can be broadly categorized into three areas: acquisition and representation of geometric object's cross-sectional information based on the NURBS data, development of a path planning system for obtaining the X-Y cross-hatch pattern and the simulation of the results with a graphical interface. This chapter provides a summary of literature in these three areas. The first and the second sections review efforts in geometric modeling and research related to the tessellation or triangulation of geometric models. The third section is a review of the curing techniques employed in rapid prototyping.

Geometric Modeling For Automated Manufacturing

Computer-aided modeling systems have evolved over the past four decades, with early research propelled by advances in Computer Aided Manufacturing (CAM). Figure 2.1 traces the evolution of this technology [12].

Until the advent of solid modeling systems, wireframe modeling supplied the base for all commercial CAD/CAM systems. The wireframe was then replaced by solid modeling as the most unambiguous method of representing solids. Boundary representation (B-Rep) and constructive solid geometry (CSG) are the two most widely used techniques in current modeling systems. The fundamental aspect of both

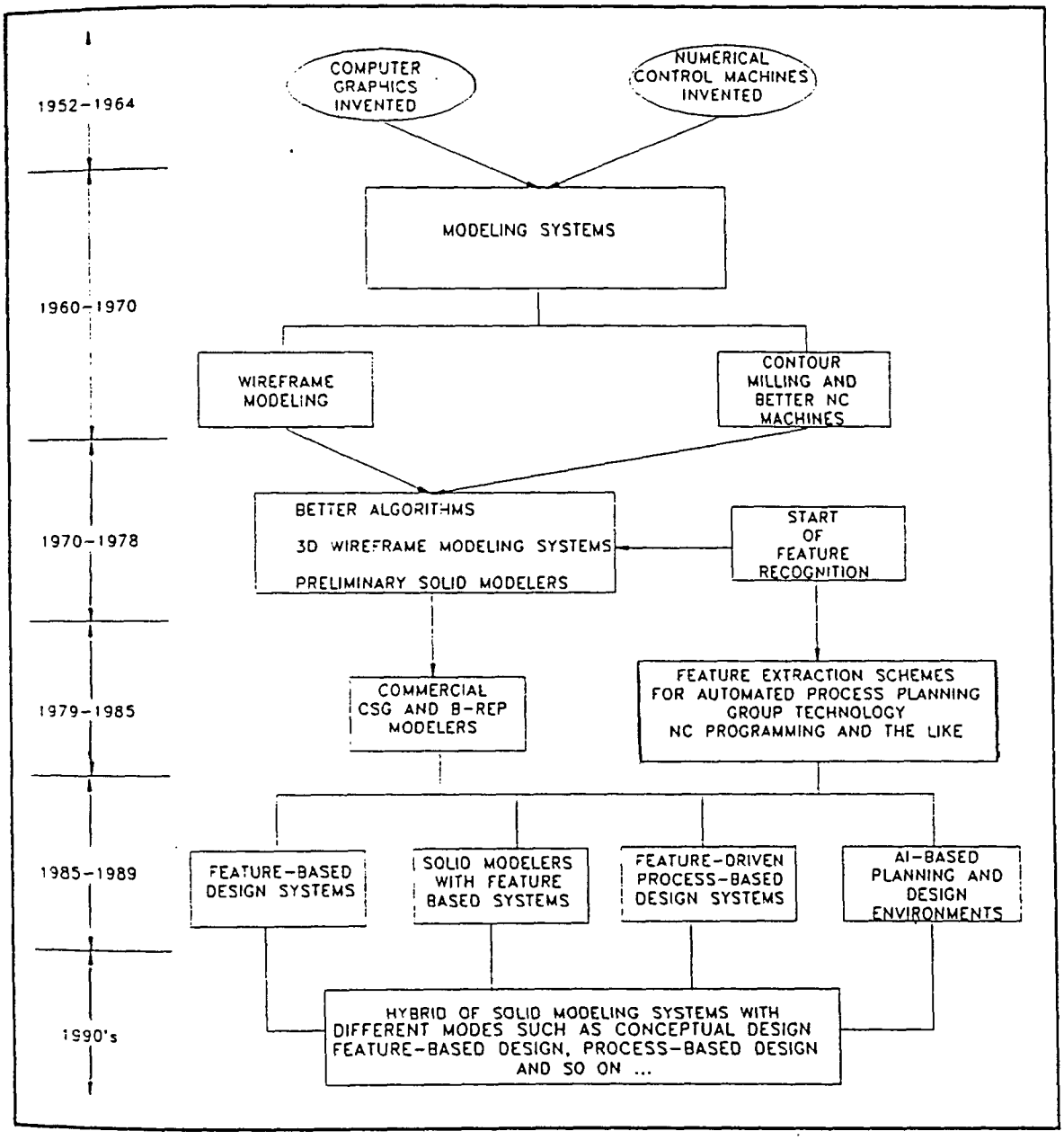

Figure 2.1: Evolution of geometric modeling technology

these cases is an effective representation of curves, surfaces and solid objects. The use of rational polynomial functions for representing curves and surfaces in CAD/CAM applications has become increasingly widespread [11]. Non-Uniform rational B-Spline (NURB) curves and surfaces have been an Initial Graphics Exchange Specification (IGES) standard since 1983 [8] and a number of commercial modeling systems exist that are based on rational Bezier or rational B-spline representations, e.g., SDRC's solid modeler, I-DEAS. This increasing popularity of NURBS is due to the following facts [11]:

1. They offer one common mathematical form for the precise representation of standard analytic shapes as well as free-form curves and surfaces.

2. They offer extra degrees of freedom (the weights)

Tessellations in Rapid Prototyping

There are several approaches to the rapid prototyping or toolless model making, but generally, the process starts with a workstation CAD design that is output to a STL format file [16], a planar faceted representation. This file is then fed to the toolless model-making machine and the object described by the STL file is sliced into cross-sections, or layers. Figure 2.2 shows the schematic of a rapid prototyping system [14]. The machine then uses one of the several techniques to deposit a material in a container in a precisely controlled manner, building the defined object one layer at a time.

The STL file defines the surface of an object as a set of interfacing triangles, similar to one that many CAD packages use to draw surfaces of on-screen images.

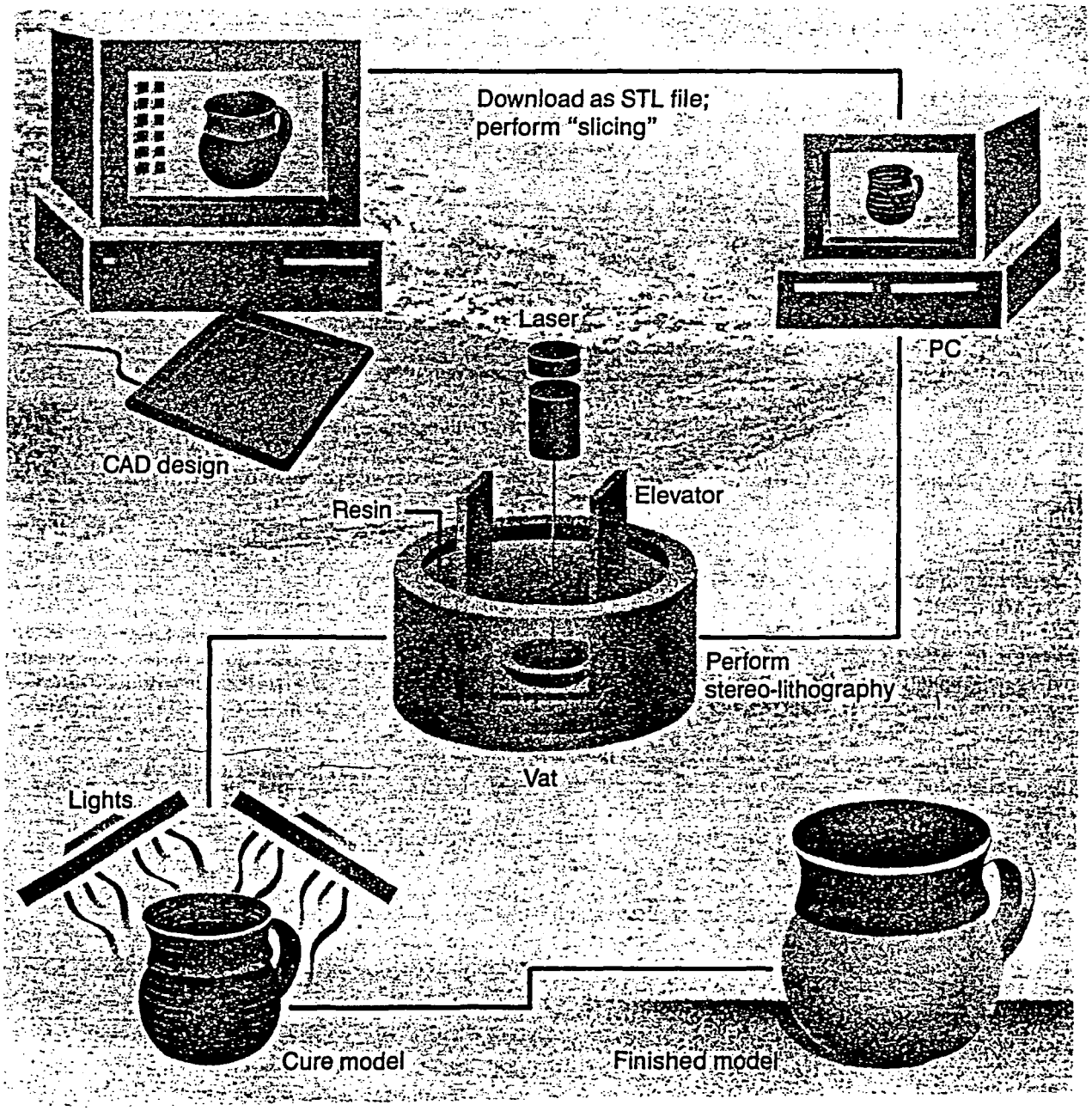

Figure 2.2: Schematic of a rapid prototyping system

Each triangle is defined with three vertices and a normal, which identifies which side faces out and which faces in. Together these triangles define a connected set of triangular facets and these facets describe the shape of the object. These facets approximate the surface of the design and are sometimes visible on the surface of the final prototype.

The uneven surface caused by the facets may require extra hand sanding or bead blasting, depending on how the part is going to be used. An STL file is often larger than the CAD file from which it was created. The actual size depends on the size of facets chosen and the shape of the part. In the current practice, higher part accuracy eventually requires higher resolution of the triangulation. Thus higher triangulation resolution also means longer processing time, correspondingly increased number of smaller triangles and a larger STL file size. These factors not only increase the cost but do not necessarily result in better STL models because the number of collapsed triangles increases as well. A triangle collapses when one or more of its sides become so small that the processing software consider them as lines. Overlapping triangles also causes stray lines or cracks in the final part [14].

Curing or Part Building Techniques

X-Y Cross-hatch Pattern

Once the CAD model is sliced into many thin layers, each layer is built in a vat of photocurable liquid acrylate resin that hardens when the laser light hits it. The laser beam is deflected in the x and y axes by galvanometer-driven mirrors so that it moves across the surface to produce a planar pattern. Part borders are scribed with the laser beam and layer's surface is formed with x-y cross-hatching passes. The part is built layer by layer from bottom up, curing one layer at a time, and finally post cured in an oven to complete the polymer solidification process.

Over the last few years, researchers have a developed a new part building technique called "Weave", which improves dimensional tolerances [4]. Typical x-y cross-hatching methods produce a rather fragile matrix of thin-walled chambers that trap liquid or semicured resin inside the part walls like water is trapped in partially frozen ice cubes in a freezer tray. Often, only 50 percent of the acrylate material is polymerized. These standard techniques, which minimize the amount of time the laser is operating, rely on the post-curing oven to achieve full solidification.

Weave Pattern

The weave method is characterized by a closely spaced (0.001 in) x-y beam hatching that is exposed so that the cure depth is slightly less than the polymer layer thickness. Thus, the x-hatching doesn't adhere to the layer below or to itself. When the y-axis hatching is drawn over x-hatching, the additional light exposure leaves little liquid resin remaining in the part. In addition, the double exposures at the

hatching intersections tack the layers together. Even though more cross-hatches are used in the Weave method than with previous hatching techniques, each individual cross-hatch has less light exposure, so the total building time is not substantially increased.

CHAPTER 3. DESCRIPTION OF THE ALGORITHMS

The 3D computer models created in the CAD system have to be sliced into a stack of 2D cross-sections for laser path planning in rapid prototyping. In commercial systems the object is tessellated (triangulated) before it is sent to the rapid prototyping system and sliced by a special software there. This results in the error/inaccuracy introduced by the triangulation to propagate through the entire process.

Overall Methodology

The problem that is to be solved as part of the overall research is to extract the information of cross-sectional profiles of the geometric model created in a solid modeling system. We approach this problem by making use of the "section cutting" and NURBS output capability built within the solid modeler. This procedure is applicable for any solid modeling system that has both these capabilities. I-DEAS is used as the example solid modeling system.

In order to arrive at a method for providing the information, the type of information that is likely to be used at a later stage needs to be known. This will help in identifying the types of parameters that are to be used. For the path planning routines, two types of information are needed which are classified as (1) KNOW WHAT, and (2) KNOW HOW. These may be defined as follows:

- KNOW WHAT specifies the important geometric information that needs to be obtained from a computer model such that it can be used by the interface routine to know what is to be path planned.
- KNOW HOW specifies the means by which the post processor can get the required path planning information. This is akin to the conventional algorithmic approach wherein numerical routines are written to perform a task.

The research initiative here is therefore to design and implement a "slicing program" that can be associated with any geometric model. The overall methodology is to provide a system that will help users to extract the information of the cross-sectional profiles based on NURBS data and to subsequently use for the generation of 2D cross-hatch patterns. The following are the steps in providing such an interface:

1. Design the part using a solid model CAD system and orient it in the way that it would be built.
2. Run the slice program such that the set of cutting planes are defined perpendicular to the build direction. Output all the curves describing the cross-sections in IGES format.
3. Transform these IGES formatted files to the interface software.
4. Obtain the 2D cross-hatch pattern for each cross-section using the intersection algorithm.
5. Verify the correctness of the solution by simulating the laser curing and the part building process using a graphical user interface.

Obtaining The Cross-sectional Profiles

The first step is to obtain the cross-sections of the object. These sections are obtained from the Boolean intersection between the object and a stack of planar faces that are appropriately spaced. The *Cross_Section_Profile* command in I-DEAS Object Modeling module [7] creates one or more profiles by performing the cross-section cut through the selected object [Figure 3.1]. One or more profiles are created and stored by this operation. The profiles corresponding to a particular cross-section are retrieved and joined together to exactly represent the cross-section of the model. This cross-sectional information is output to an IGES file with the *Rational_B-Spline* option. IGES files are those which conform to the Initial Graphics Exchange Specifications [8]. Figure 3.2 shows the representation of rational curve entities in IGES format.

The above operations have to be repeated for every cross-section. These operations can be automatically executed using the I-DEAS *program file* capability. A program file is an external file that records the commands or interactions with I-DEAS. Once created, it can be executed to repeat the commands recorded in the file.

The object is first oriented in the direction in which it would be built. A slice program is executed which slices the model into various cross-sections (allowing varying thickness). The curves describing each cross-section are stored separately for path planning individual cross-sections. Figure 3.3 shows the result of this operation. The slicing program is listed in Appendix C.

The reasons for using I-DEAS, a solid modeler to extract the cross-sectional information are:

The *Cross_Section* command creates one or more stored profiles by performing a cross section cut through the selected object.

/Create | Special_Techniques | Cross_Section_Profile

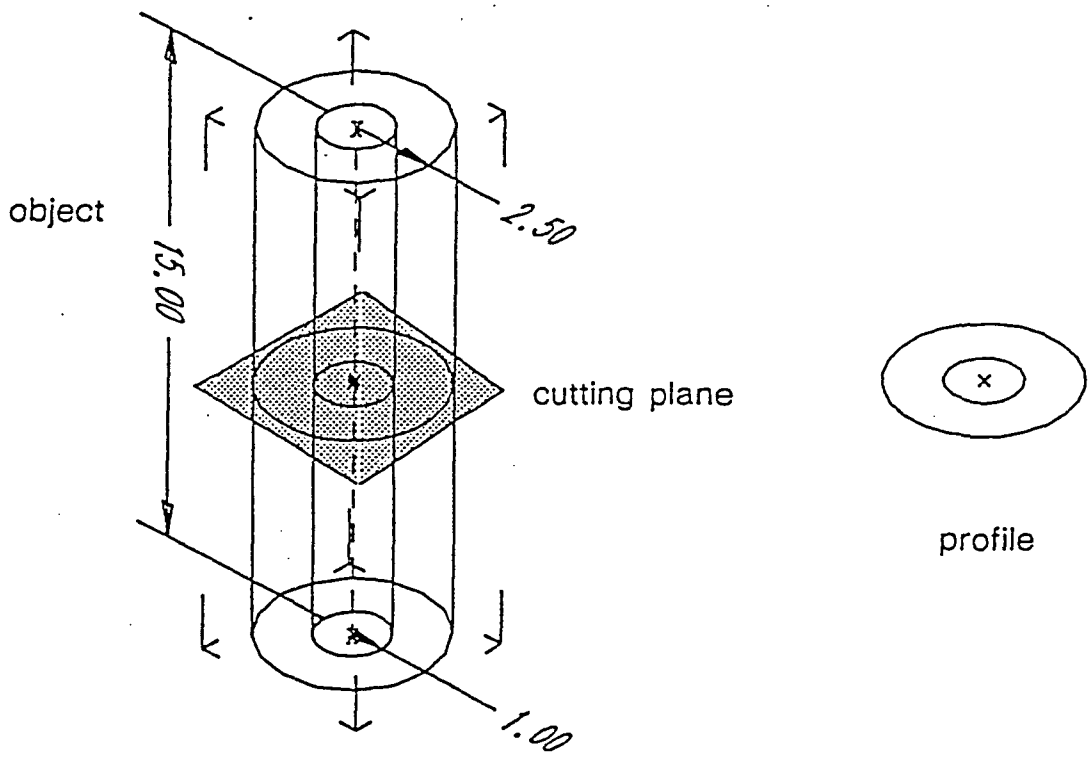

Figure 3.1: Creating a profile from an object

Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
126	⇒	< n.a. >	#, ⇒	#, ⇒	0, ⇒	0, ⇒	0, ⇒	??????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
126	#	#, ⇒	#	#				#	D # + 1

Note: Valid values of the Form Number are 0–5.

Parameter Data

Index	Name	Type	Description
1	K	Integer	Upper index of sum. See Appendix B
2	M	Integer	Degree of basis functions
3	PROP1	Integer	0 = nonplanar, 1 = planar
4	PROP2	Integer	0 = open curve, 1 = closed curve
5	PROP3	Integer	0 = rational, 1 = polynomial
6	PROP4	Integer	0 = nonperiodic, 1 = periodic
Let N = 1+K-M and A = N+2*M			
7	T(-M)	Real	First value of knot sequence
⋮	⋮	⋮	
7+A	T(N+M)	Real	Last value of knot sequence
8+A	W(0)	Real	First weight
⋮	⋮	⋮	
8+A+K	W(K)	Real	Last weight
9+A+K	X0	Real	First control point
10+A+K	Y0	Real	
11+A+K	Z0	Real	
⋮	⋮	⋮	
9+A+4*K	XK	Real	Last control point
10+A+4*K	YK	Real	
11+A+4*K	ZK	Real	
12+A+4*K	V(0)	Real	Starting parameter value
13+A+4*K	V(1)	Real	Ending parameter value
14+A+4*K	XNORM	Real	Unit normal (if curve is planar)
15+A+4*K	YNORM	Real	
16+A+4*K	ZNORM	Real	

Figure 3.2: Representation of curves in IGES format

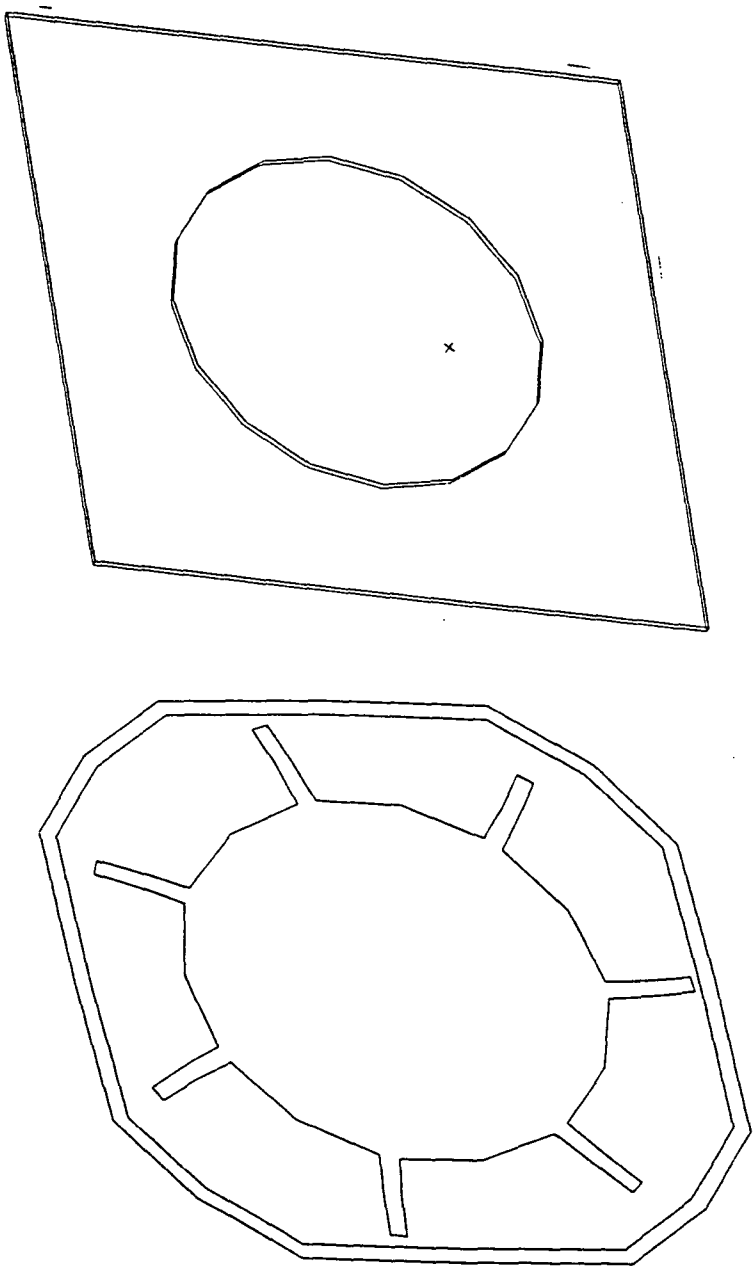

Figure 3.3: Cross-sectional profiles

1. Its representational scheme is based on NURBS.

2. It supports the "section cutting" capability.

3. Its slicing operation can be programmed to automatically slice the model and store the cross-sections separately in IGES formatted files.

4. It was the modeling system readily available for research.

After slicing, the geometric information has to be reduced to on/off toggle points for the laser.

Vector Scanning or Path Planning (2D)

Many geometric algorithms include tests for intersection between geometric entities. For example, identification of the interior and skin-fill areas for laser curing applications involves intersection operations. The intersection between the cross-section and the cross-hatch grid identifies the vectors to be scanned by the laser scanner system. The object boundaries for the laser are quickly found from the perimeter of the cross-sections. Similarly, the grids for the laser path planning are defined by the intersection of the perimeters of the cross-sections with the grid of line segments.

The principle of vector scanning is that any line can be broken into its x and y cartesian components and that any 2D shape can be drawn on the polymer surface by coordinating the motion of the laser scanner system. The intersection points can be used to signal the laser to turn off during the nonwrite movements. This action is analogous to the pen-up/pen-down function of a pen plotter.

The algorithm reads in each cross-sectional information and produces a curve pattern that defines the solid's boundary. The boundary is then filled with a preselected grid pattern. Finally, these grids from the slice-and-fill algorithm are divided into small intersecting segments needed to step the laser scanner system smoothly along the solid area leaving the hole portions uncured.

Method of Ray Casting

Ray casting is a method normally performed by intersecting a ray (a semi-infinite line) against the curve/surface elements of a geometric model [13]. In this case, this method is used to differentiate between solid and hole boundaries. Basically a two dimensional grid of semi-infinite lines are chosen that covers the cross-sectional profile being path planned. A ray (defined by its starting point q and its direction u) is cast along each of the lines on the grid and all the intersections between this ray the cross-sectional profile are obtained.

Computing Intersections

Intersection is a fundamental process in computational geometry, needed to build and interrogate models of complex shapes in the computer. Intersection computations are needed to evaluate set operations on the cross-sectional profiles in creating cross-hatch patterns of complex boundaries.

In this case, the problem is defined as follows:

Given a cross-sectional profile (or curve) and an equation of a line segment (a pair consisting of a point q and a direction u), obtain all the intersections along the path of a straight-line ray $\vec{r}$ from q in the direction of u with the given profile.

Subroutines from the DTNURBS library (general spline evaluation programs developed by the Boeing Computer Services and distributed by the David Taylor Research Center) are used to evaluate these intersections and the implementation details are discussed in Chapter 4. These routines use the interval Newton method to compute the intersections. A brief description of this method is given in Appendix D.

Obtaining the Cross-hatch Pattern

Having found the intersections points, they are classified as on/off toggle points for the laser scanner system to traverse the cross-sectional contours. Here it is assumed, that the starting point of the ray is outside the cross-sectional profile. Apart from the singular cases when the ray hits the cross-sectional profile in a single point or goes along its boundary curve, it will intersect the cross-sectional profile at an even number of points, denoted by $t_1, t_2, ..., t_n$ in cartesian space. With k = 0, ..., n/2, the ray is within the solid area for the points $t_{2k} < t < t_{2k+1}$ and outside the solid area for the points $t_{2k+1} < t < t_{2k+2}$. When the number of intersections is odd, for k = 0, ..., n-1, the ray is within the solid area for $t_k < t < t_{k+1}$ and outside otherwise. Unlike the situation in which ray casting is used to generate shaded images, where only the first actually visible intersection point is of interest, here all intersections need to be evaluated for identifying "internal" details. This information is utilized by the laser scanner system to turn on/off the beam according to whether the point is inside the solid area or outside.

An important parameter in this ray casting technique is the size of the grid (or the semi-infinite line) [18]. If this is too small, it is possible that the intersections

with some of the curves will not be detected. Additionally, a large grid size will result in time consuming computations that are not necessary. This implementation currently uses a fixed grid size which is assumed to be large enough to detect all the intersections. It may be possible to detect the grid size automatically based on the maximum dimension of the cross-sectional profile or the model.

Summary

This chapter provides for the first time an interface routine for geometric models based primarily on NURBS. The procedures have been explained and representational issues outlined. The implementation of this method is outlined in Chapter 4. Also, generating a building pattern of vectors that will correctly construct the part is verified using a graphical simulation interface.

CHAPTER 4. SOFTWARE IMPLEMENTATION

The algorithms discussed in the previous chapter have been implemented in C language on both DEC and SGI hardware platforms. A module based approach is used to achieve the implementation. The various stages of the algorithm are implemented as individual modules to enable easy development and debugging.

The main program reads the necessary input from an IGES file whose name is automatically generated by a built-in I-DEAS capability. The information obtained from an IGES file is in the NURBS format. The NURBS representation is discussed in Appendix B. This algorithm assumes that all the feasible cross-sectional profiles are planar and therefore only the curves defining the cross-section are read. A planar curve is a spline curve that lies in a plane.

Curve Evaluation

The following sections describe the subroutines used to read and evaluate the B-Spline curves and also to compute the intersection of curves with line segments.

Reading an IGES Formatted File

This subroutine (DTRDIG) converts the spline vector for an object from the IGES format (a two dimensional array) to DTRC format, a one dimensional array

(C-array) used by DTNURBS.

The cross-sectional profiles are read one at a time and all the parameters required to define a tensor product spline curve are stored in a single one-dimensional array of double precision values. These representations are based on B-Splines. The parameters are the numbers of independent and dependent variables, the degrees and the numbers of B-Spline coefficients, the knots in each variable, and the B-Spline coefficients. The array can then be input to any of the spline evaluation subprograms for further processing. The data structure used to represent the NURB curve is shown in Table 4.1 and a sample spline array is shown in Table 4.2. The data structure contains a pointer to the order of the curve, the number of control points, a pointer to the array of knots and a pointer to the set of control points and weights (i.e., B-spline coefficients).

Evaluation of a Spline

This subroutine (DTSPVL) computes the value of a rational spline given by the data in the spline definition C-array and a parameter value input.

All the curves describing a cross-section are evaluated and it is filled in using the intersection algorithms for 2D path planning for the laser beam. The cross-hatch pattern is obtained by computing the intersections between a mesh of line segments and the curves defining the cross-section.

Table 4.1: Structure for representing a NURB curve

Index	Name	Description
1	1	number of parametric(independent)variables
2	-2^a	number of dimensions (dependent variables)
3	k	order of the spline (equal to degree + 1)
4	N	number of B-Spline coefficients
5	$jspan^b$	index of last span located
Knots		
6 to 5+N+k	$\vec{z}$	knot array
Coefficients		
6+N+k to 5+2N+k	a_j	B-Spline coefficients numerator
6+2N+k to 5+3N+k	w_j	B-Spline coefficients denominator

[a] The sign of the number of dependent variables is used to determine whether the function is a polynomial spline or a rational spline. if the number $m > 0$ the function is a polynomial spline with m dependent variables. If $m < 0$ it is a rational spline with $|m| - 1$ dependent variables. The $|m|^{th}$ coordinate is the denominator.

[b] The parameter *jspan* is the index of the knot interval in which the last evaluation point was found. It is, strictly, not a true spline parameter, and hence it is not necessary for defining the spline. However, *jspan* is maintained because of its utility in speeding up the execution of evaluations.

Table 4.2: Example of a spline C-array

Index	Data
c[0]	1
c[1]	1
c[2]	4
c[3]	4
c[4]	b^a
c[5]	0
c[6]	0
c[7]	0
c[8]	0
c[9]	1
c[10]	1
c[11]	1
c[12]	1
c[13]	0
c[14]	0
c[15]	0
c[16]	1

[a] This is the representational array for a cubic $s(t) = t^3$ on the interval [0,1]
Here, b represents the *jspan* parameter. For a rational spline curve with two dependent variables c(2) will be equal to -3.

Path Planning (2D)

Once the solid model is converted into an ordered set of 2D cross-sections, each cross-section is intersected with a preselected cross-hatch grid of line segments. This generates appropriate drawing vectors to be traversed by the laser scanner system.

Computing Curve-Line Intersection

The subroutine (DTCLXT) computes the parametric values of all points and intervals of intersection between a curve and a line. This routine assumes both the objects are in the plane.

The line is provided as an array of coefficients (a,b,c) for a defining equation of the form ax + by = c. A real valued spline function is constructed by substituting the curve's coordinate function in the equation for the line. The zeroes of this spline function are then found by using DTSZER subroutine.

Obtaining the Cross-hatch Pattern

Having found the intersection points, they are be classified as points inside/outside the solid area. The intersection points along any direction are sorted in ascending order based on either x or y component. Using these data and the method of ray casting, the laser curing path is defined as a cross-hatch grid of line segments. The skin fill (or solid) areas are shown as grid patterns and the interior (or hole) portions are left blank.

Summary

The working of the software is demonstrated using two examples on both DEC and SGI hardware platforms using HOOPS and GL graphics interfaces respectively. The program was run for two different models and the results are shown in Chapter 5. The examples show the laser scanner path as a cross-hatch pattern on the cross-sectional profiles.

CHAPTER 5. EXAMPLES

Examples are provided to explain the method and how it works with different computer models that have engineering applications. Here it is assumed that the solid modelers usually support the "section cutting" capability.

Example 1: A Simple Geometry: Model of a Hub

2D path planning was attempted for the model of a Hub [figure 5.1] to highlight the method. The model under consideration was oriented with z-axis up. This is a rear Hub designed for a high speed human powered vehicle. This model was created using the solid modeling and feature based modeling techniques of I-DEAS. As earlier stated, sectioning was performed inside the solid modeler and figure 5.2 shows some of the cross-sectional profiles. These profiles were transferred to a IGES formatted file.XS Figure 5.3 shows individual profile cross-hatch pattern and figure 5.4 shows the part build-up process.

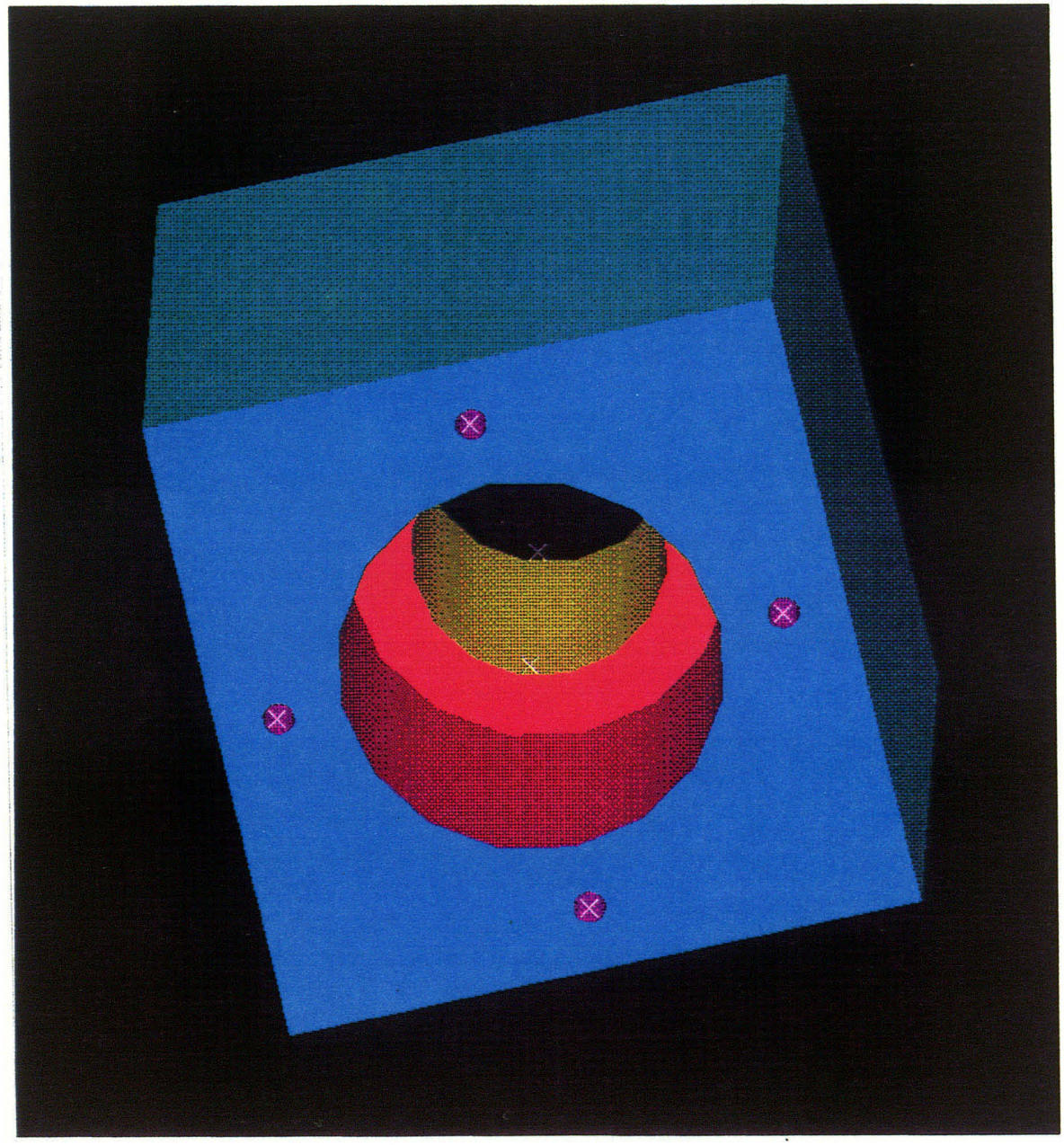

Figure 5.1: A model of a hub

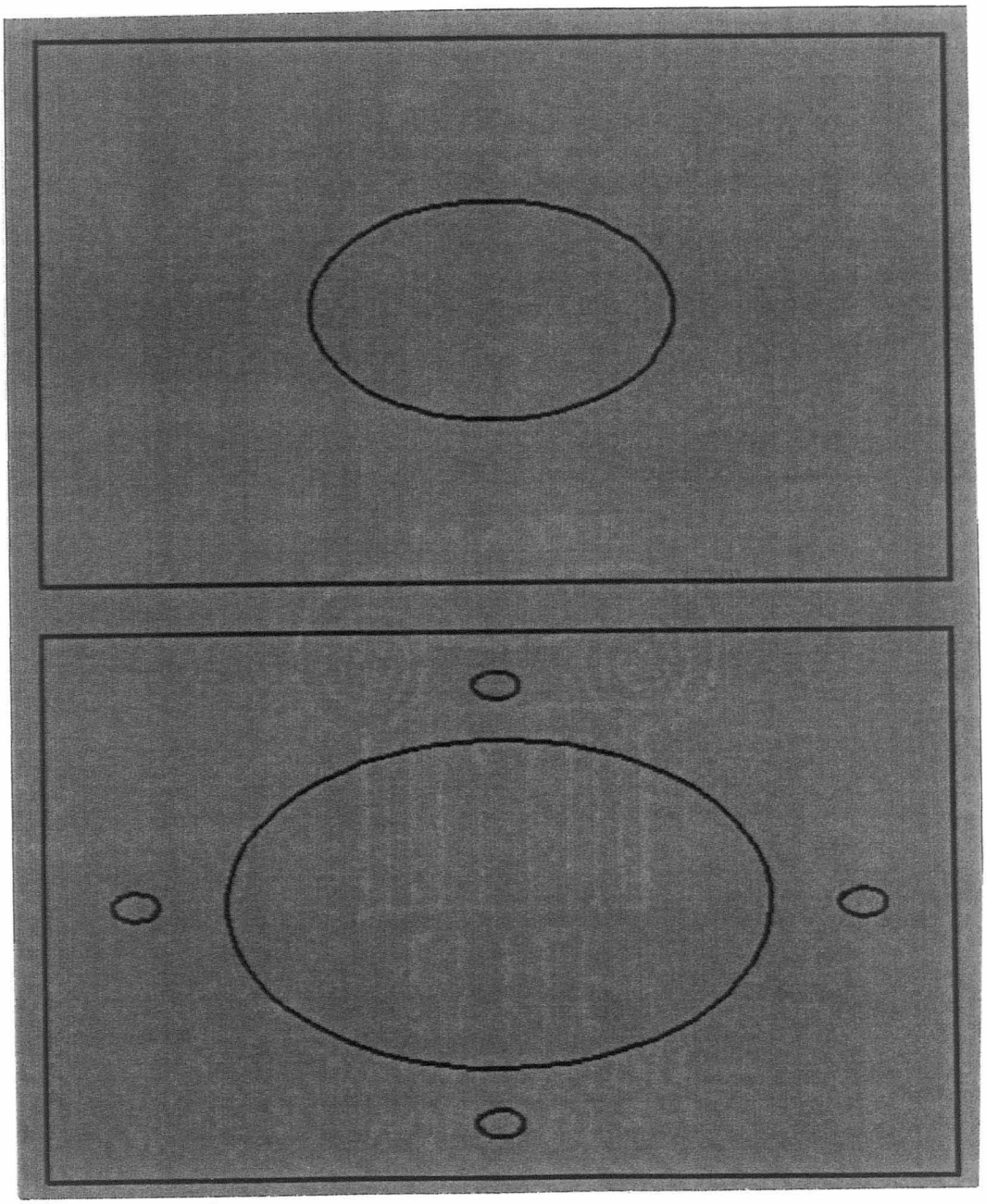

Figure 5.2: Cross-sectional profiles

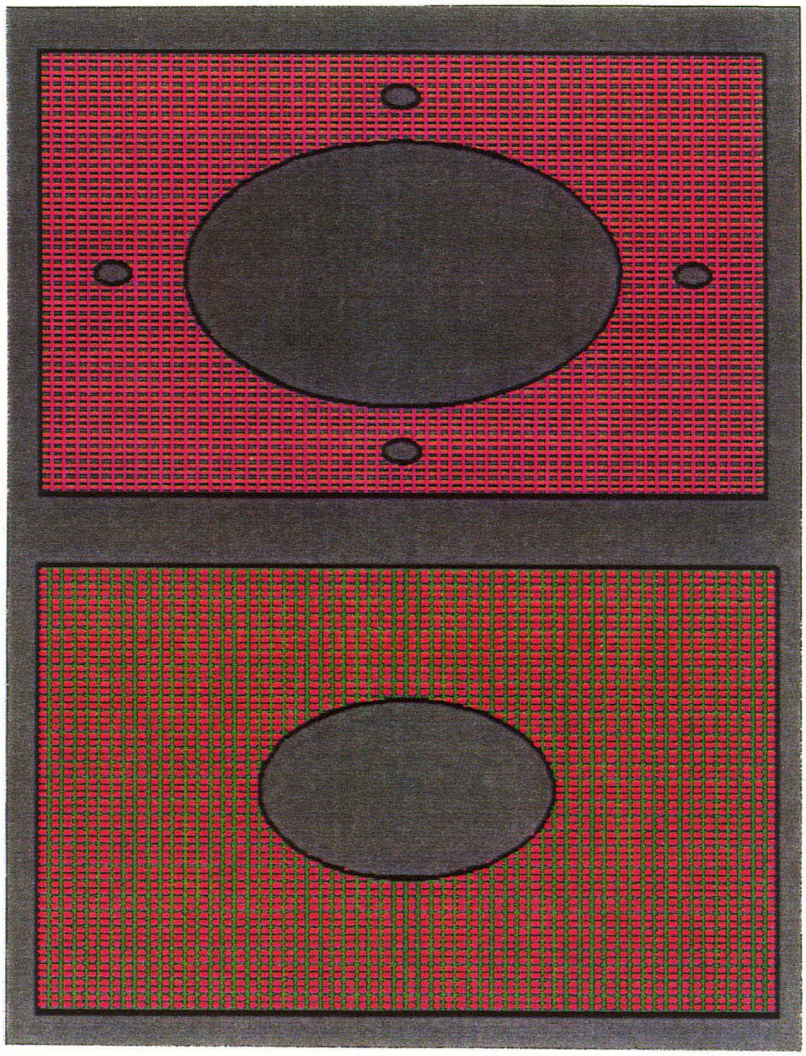

Figure 5.3: 2D cross-hatch pattern for individual profiles

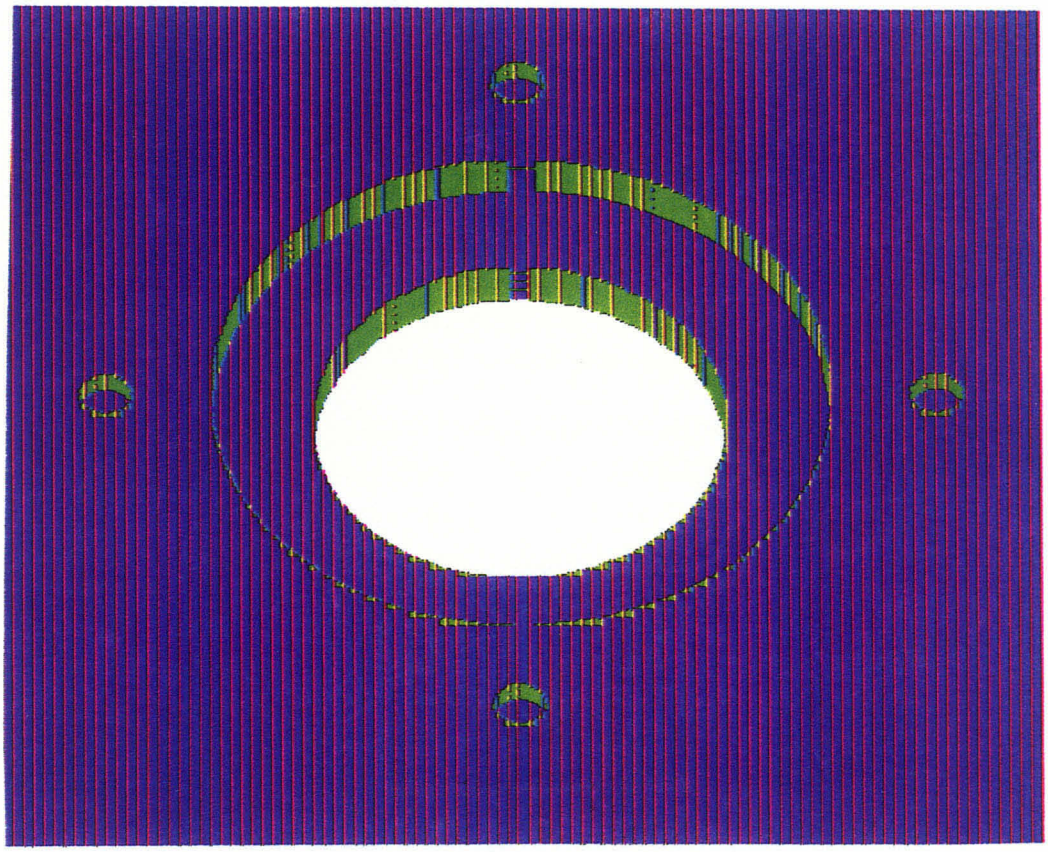

Figure 5.4: Part build-up process

Example 2: A Complex Geometry: Model of a Fan and Shroud

A relatively complex geometry was created using some of the advanced surface and object creation methods in I-DEAS. The example is a model of a high performance cooling fan for electronic equipment [figure 5.5]. The shroud around the fan is modeled from cross-section and are used to "skin" the solid object. This shroud illustrates a classic case of transitioning from a round section around the fan to a square mounting flange. The fan blades were modeled using different surfacing techniques. Seven blades are then combined with the rotor to make the completed fan. Figure 5.6 shows the cross-sectional profiles created by slicing operation. Figures 5.7 and 5.8 show the 2D curing pattern and the part building-up process. This figures also show the ability of the software to vary the mesh size. This is especially useful where a fine mesh is needed for high surface finish.

Summary

These examples have shown the use of the novel NURBS based path planning method for Rapid Prototyping. A need was established for a precise representation of the curves and surfaces of the object for improved tolerance and surface finish capabilities. The method based on NURBS representation was outlined and implemented in different hardware platforms which adds to the portability of the code. Traditional Newton-Raphson method was used to obtain the curve/line intersections that is typical of intersection algorithms. A special case of Ray casting method was however added to differentiate between solid/hole boundaries. This robust mathematical technique has great potential to be applied in future rapid prototyping systems.

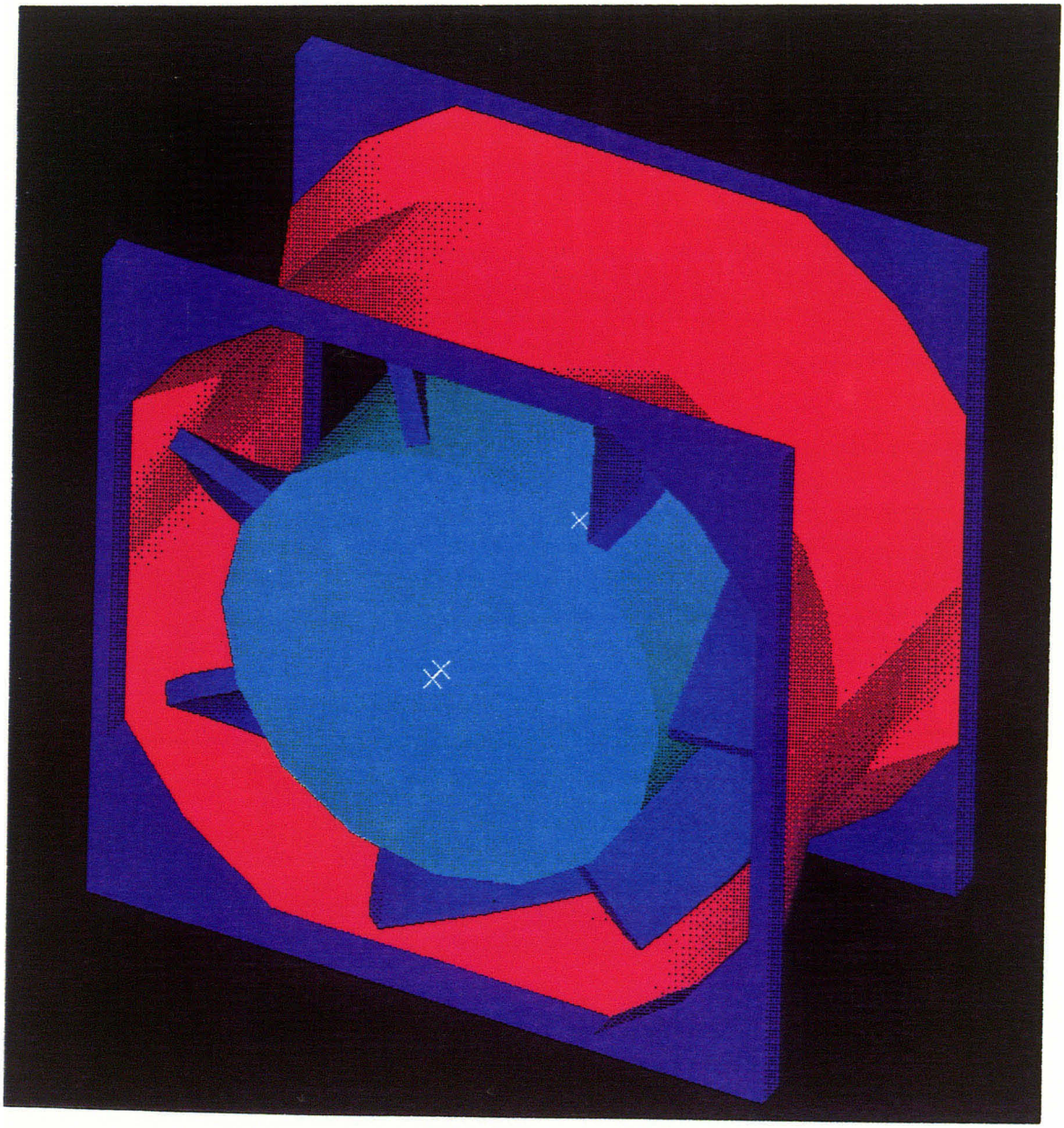

Figure 5.5: A model of a fan and shroud

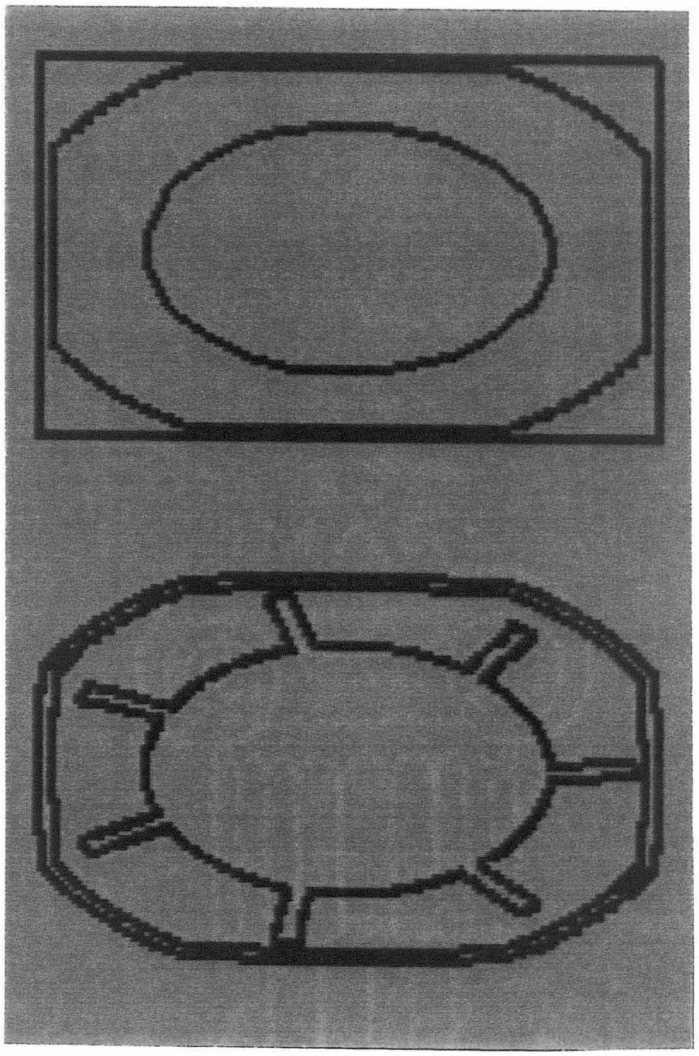

Figure 5.6: Cross-sectional profiles

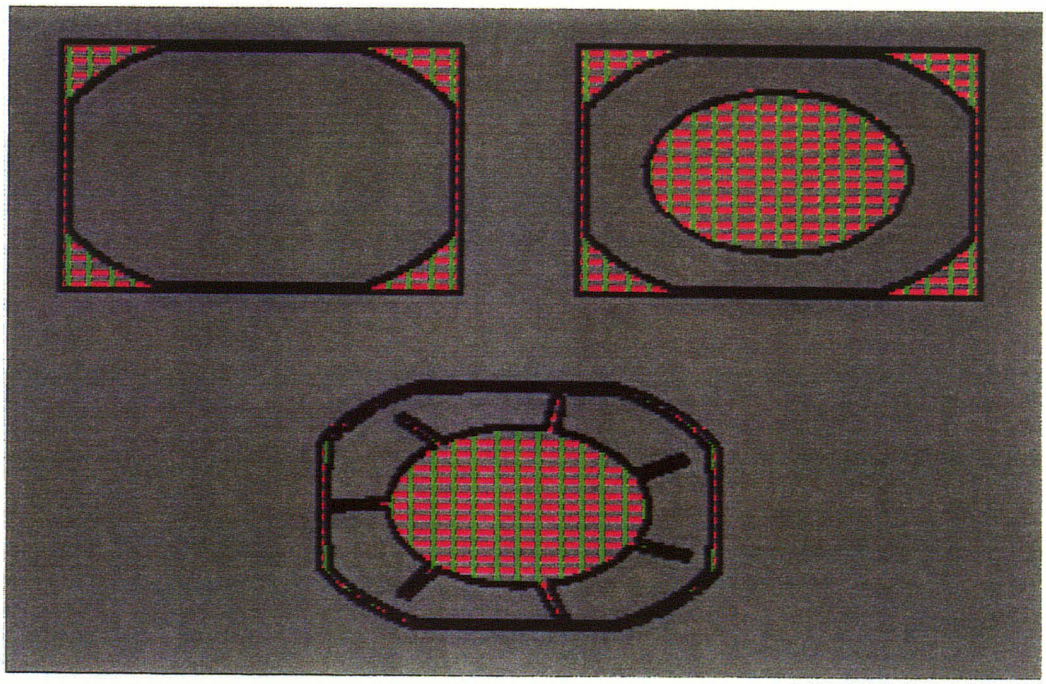

Figure 5.7: 2D cross-hatch pattern for individual profiles

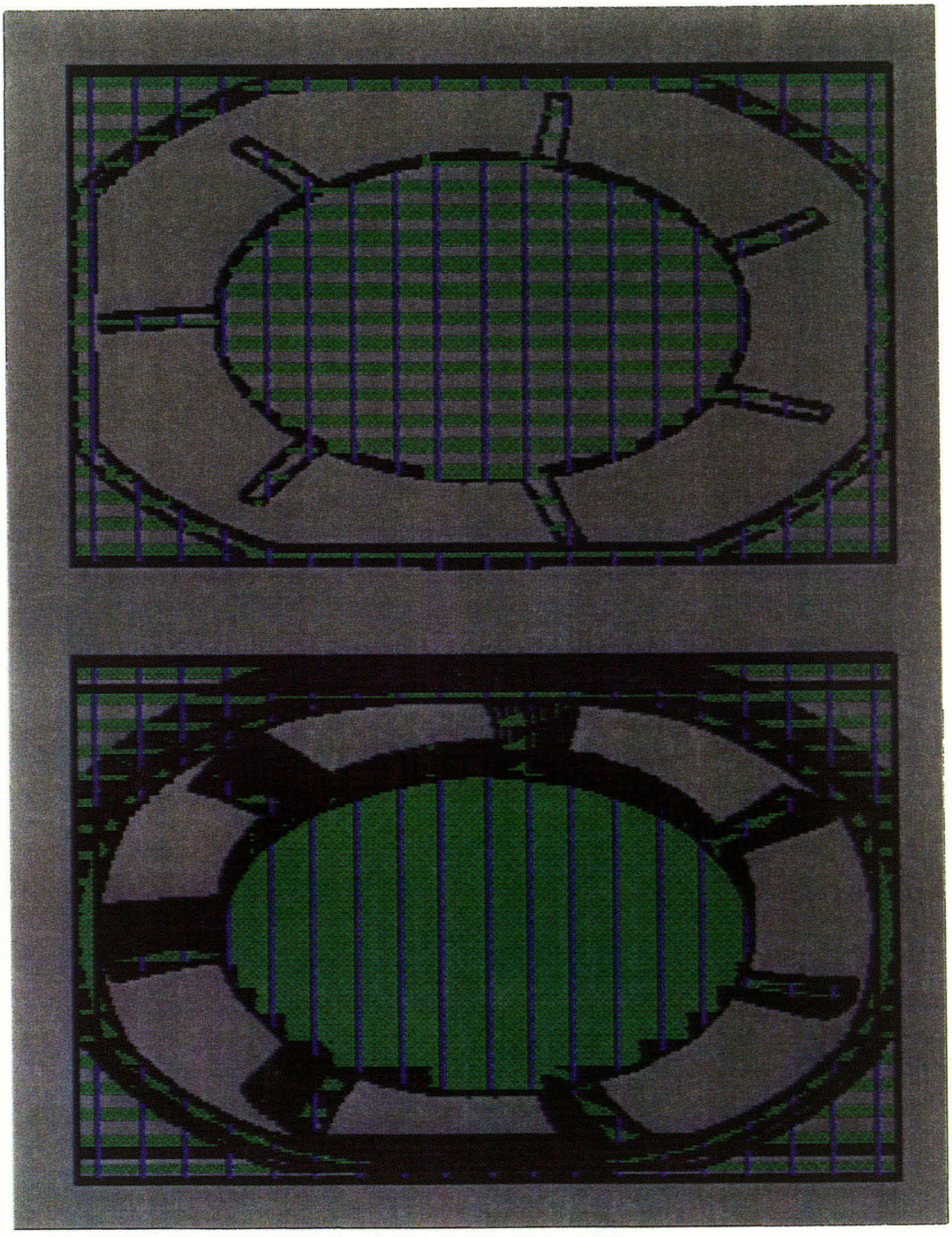

Figure 5.8: Part build-up process

CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS

The objective of this research was to develop a computer interface for rapid prototyping by utilizing the common underlying representational structure of the objects for both slicing (or section cutting) and path planning (2D). The purpose of this Chapter is to summarize the conclusions drawn from the development of such an interface and propose recommendations for future research.

Summary

Rapid automated prototyping is complete when there exists a system for off-line creation, slicing and path planning of the 2D cross-sectional profiles of objects, and an online laser post processing system with feedback to control the curing process. This work concentrated on the first part - that of generating path planning information for the laser post processor off-line based on precise CAD description. This research provides a framework on which other interfaces can be built to close the feedback loop and make it completely a CAD based automatic prototyping system. Important contributions by way of this research can be summarized as:

1. Current commercial rapid prototyping systems do not make use of the representational geometry of the solid modeling systems. This research for the first time has implemented a routine that uses the same underlying geometric represen-

tation (NURBS) for object creation, slicing and path planning the individual cross-sections.

2. Geometry information is extracted from the modeling system and transferred to the interface routines through IGES files. This increases the usability of the software with all the commercial modeling packages that has "section cutting" and IGES output capabilities.

3. The path planning (2D) is robust mathematical technique to obtain the cross-hatch pattern for the laser post processor. Various profile geometry normally encountered in day-to-day models were tested to verify the method's ease and power.

Thus, the advantages of developing such a NURBS based computer interface using precise geometric representations can be summarized as follows:

- Allows finer detail by operating directly from the CAD design.
- The software used a geometrical mathematical model rather than approximated information (based on faceted models) to steer the post processor.
- The software is compatible with most commercially available CAD equipment and solid modeling software that supports IGES formatted output capability.
- Results in the reduction of computation time as there are only fewer data conversions involved.

Recommendations For Future Work

Various aspects of the research that can be explored based on this work are suggested in this Section. These are based on the experience gained during the development of the computer interface. These additions to the interface will help in the creation of a truly automatic prototyping system for rapid prototyping based on the precise representation of geometric modeling packages. This will eventually help in the use of rapid prototyping techniques where high tolerance and surface finish capabilities are required.

Detailed Information Extraction

This research proposed a basic technique to extract the precise cross-sectional profile (or curve) information of the object from a solid modeler. However, to support precise rapid prototyping through off-line programming, extraction of cross-sectional surface information should be developed. This will assist in the path planning with richer information regarding the solid/hole boundaries. This will also result in the better utilization of the IGES interface formatting capability.

Laser-Robotic Post Processing

In vector scanning and post processing galvanometers are the heart of the scanning system and the speed and inertia characteristics of the scanning galvanometers limit the time required to start, stop and settle the motion at each beam destination. Another approach could be to use robot and fiber optics assembly to control the cure rate and energy sharing. The robot can be driven through automatic programming using the cross-sectional data. Fiber optics will be useful in delivering greater

quantities of energy in short intervals and thus establishing more on-target power.

Integration of Prototyping Process Knowledge With CAD

It is uniformly seen that representation of information in a CAD-based system is currently used only to drive an application system or help in the planning and programming of elements in the manufacturing domain. Current modeling systems do not provide a means to model the process. Various systems have been created to use the facilities of a modeling system for application purposes and they exist as stand alone systems. Instead, if provision is created in CAD systems in the form of a shell such that it accommodates different knowledge representation schemes, then automatic planning and programming in different domains of manufacturing can be created easily. This may include the rapid prototyping process knowledge and laser path planning information in addition to the common underlying NURBS representational scheme for both the geometric and process models.

REFERENCES

[1] Agarwal, P.K. 1991, *Intersection and Decomposition Algorithms For Planar Arrangements*, Delmet Publishers Inc., Albany, NY.

[2] Ashley, S. 1991, "Rapid Prototyping Systems," *Mechanical Engineering*, Vol. 4, pp. 34-43.

[3] Chazelle, B. and Guibas, L. 1989, "Visibility And Intersection Problems In Plane Geometry," *Discrete and Computational Geometry*, Vol. 4, pp. 551-581.

[4] Guessel, T. 1989, "Affordable Prototypes Made Without Tooling," *Modern Plastics*, Vol. 66, no. 3, pp. 84-86.

[5] Hawkins, J.L. 1990. "The Wonder of B-Splines, A Beginners Guide To The Mystery And Power of B-Splines and Their Variations," Dept. of Mech. Engg., Michigan State University, E. Lansing, MI.

[6] Heller, T. 1990. "Software Enables Laser Stereolithography," *Laser Focus World*, Vol. 7, pp. 57-58.

[7] IDEAS 1991. Solid Modeling User's Guide, SDRC Corporation, Milford, OH.

[8] IGES, 1991. Initial Graphics Exchange Specification, American National Standard, Version 5.1, International TechneGroup Inc., Milford, OH.

[9] Nielson K.L. 1965. "Methods in Numerical Analysis," The Macmillan Company, New York.

[10] Nutt, K. 1991. *The Selective Laser Sintering Process*, DTM Corporation, Austin, TX.

[11] Piegel, L. and Tiller, W. 1987. "Curve and Surface Constructions Using Rational B-Splines," *Computer Aided Design*, Vol. 19, no. 9, pp. 485-497.

[12] Requicha, A.A.G., and Voelcker, H.B., 1982, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, Vol 2. no.2, pp. 9-24.

[13] Sweeny, M.A.J., and Bartels, R.H., 1986, "Ray tracing free-form B-Spline surfaces," *IEEE Computer Graphics and Applications*, Vol. 6, no. 2, pp. 41-49.

[14] Wohler, Terry
~~Terry, T.W.~~ 1992, "CAD Meets Rapid Prototyping," *Computer Aided Engineering*, Vol. 2, pp. 66-74.

[15] Varady, T. and Gaal, B., 1990, "Identifying Features in Solid Modeling," *Computers in Industry*, Vol. 14, no. 1-3, pp. 43-50.

[16] Wood, L. 1991. "Desktop Prototyping," *BYTE, Action Summary*, Vol. 4, pp. 137-142.

[17] Wu, D.S. 1990, "Optical Scanner Design Impacts Rapid Laser Prototyping," *Laser Focus World*, Vol. 11, pp. 99-106.

[18] Zheng, J.L., and Millham, C.B., 1991, "A Linear Programming Method for Ray-Convex Polyhedron Intersection," *Computers and Graphics*, Vol. 15, no. 2, pp. 195-204.

APPENDIX A. GLOSSARY OF TERMS

Abbreviations:

B-Rep	:	Boundary Representation
CAD	:	Computer-Aided Design
CAM	:	Computer-Aided Manufacturing
CSG	:	Constructive Solid Geometry
DTRC	:	David Taylor Research Center
NURBS	:	Non-Uniform Rational B-Splines
STL	:	Stereolithography

Terms:

Cartesian Space: Generally a two or three-dimensional rectangular space defined by mutually perpendicular Cartesian axes, normally denoted as x and y, or x, y, and z.

C-array: It is a one dimensional array contained DTRC format B-Spline data.

APPENDIX B. NON-UNIFORM RATIONAL B-SPLINES (NURBS)

The following definition of NURBS is from Piegl's survey [1991].

The two major ingredients of a Non-Uniform Rational B-Splines are rational and B-splines. A NURBS curve is a vector-valued piecewise rational polynomial function of the form [11]:

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,p}(u)}{\sum_{i=0}^{n} w_i N_{i,p}(u)} \tag{B.1}$$

where the w_i are the so-called weights, the P_i are the control points (just as in the case of nonrational curves), and $N_{i,p}(u)$ are the normalized B-spline basis functions of degree p defined recursively as

$$\begin{aligned} N_{i,0}(u) &= 1 \ if u_i <= u_{i+1} \\ &\quad 0 \ otherwise \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \\ &\quad \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned}$$

(B.2)

where u_i are the so-called knots forming a knot vector. The degree, number of knots, and number of control points are related by the formula m = n + p + 1.

$$U = u_o, u_1, ..., u_m$$

(B.3)

A NURBS surface is the rational generalization of the tensor product nonrational B-spline surface and is defined as follows:

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}P_{i,j}N_{i,p}(u)N_{j,q}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}N_{i,p}(u)N_{j,q}(v)}$$

(B.4)

where $w_{i,j}$ are the weights, $P_{i,j}$ form a control net, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the normalized B-splines of degree p and q in the u and v directions, respectively, defined over the knot vectors

$$\begin{aligned} U &= [0, 0, ..., 0, u_{p+1}, ..., u_{r-p-1}, 1, 1, ..., 1] \\ V &= [0, 0, ..., 0, v_{q+1}, ..., u_{s-q-1}, 1, 1, ..., 1] \end{aligned}$$

(B.5)

where the end knots are repeated with multiplicities p+1, and q+1, respectively, and r = n + p + 1 and s = m + q + 1. Although the surface was obtained by

generalizating the tensor-product surface form, a NURBS surface is, in general, not a tensor-product surface.

APPENDIX C. SLICE PROGRAM

The slice program written in I-DEAS object module is as follows:

K : #ECHO ALL

K : #POSI = "N"

K : #H = 0.0

K : #SL = 0.01

K : #PRO = 1

K : #IF(POSI EQ "N")THEN #INPUT YES_NO "Have you positioned the object?"
POSI

K : #IF (POSI EQ "N")THEN #GOTO DISPQUIT

K : #INPUT 'Enter the height of the object (¡10 inches)' H

K : #IF ((H LE 0.0) OR (H GT 10.0))THEN #GOTO DISPQUIT

K : #TOPP:

K : CR SPE CR Z SL

K :

K :

K : TA CG MA PR G ? PRO AU

K : !!MO P J ? PRO+1

K :

```
K :
K : MA PR STO PRO+2 PRO+2
K : #ON_ERROR PAUSE
K : MA WR CO I P P G R D E ? PRO+2
K : PRO
K :
FS:
K : 'new'+PRO;
K : YES
FD:
K :
K : #ON_ERROR PAUSE
K : TA O
K : #SL=SL+0.995
K : #IF (SL GE H) THEN #GOTO DISPQUIT
K : #PRO = PRO+3
K : #GOTO TOPP
K : #DISPQUIT:
K : #EXIT
E : *** END OF PROGRAM FILE ***
```

APPENDIX D. NEWTON-RAPHSON METHOD

Given a parametric surface definition P(u,w), evaluation of a point (or derivative) is straight forward (*i.e*), given $(u, w) \rightarrow$ P $\rightarrow (x, y, z)$. But, to compute intersections we require the inverse operation (*i.e*) given $(x, y, z) \rightarrow P^{-1} \rightarrow (u, w)$. A closed form solution for P^{-1} is impossible. So, this problem is solved numerically. This method [9] for the solution is based on the method of successive approximations.

For the simple case of a 2D parametric curve, given P(u) and $P^u(u)$, find the value of u where x = x^*. [Figure D.1]

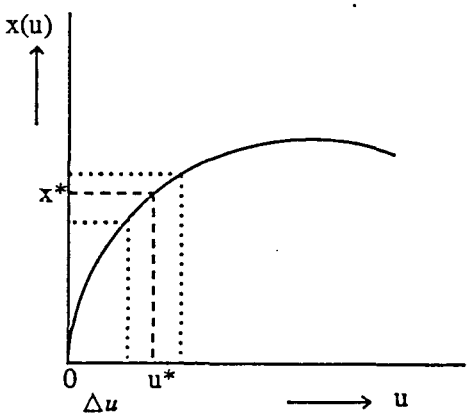

Figure D.1: The newton-raphson method

Let (u_0) be the initial approximate value. The method seeks to obtain a correction, Δu, on u_0 so that the corrected value will be

$$u_1 = u_0 + \Delta u \tag{D.1}$$

This can be expanded by Taylor's theorem for x(u) as

$$x(u + \Delta u) = x(u) + \Delta u x^u + \frac{\Delta u^2}{2} x^{uu} + ... \tag{D.2}$$

which when truncated to a first order approximation yields

$$x(u + \Delta u) \approx x(u) + \Delta u x^u \tag{D.3}$$

In order to find the correction Δu, so that $x(u + \Delta u) = x^*$,

$$\begin{aligned} x^* = x(u + \Delta u) &= x(u_0) + \Delta u x^u(u_0) \\ (or) & \\ \Delta u &= \frac{x^* - x(u_0)}{x^u(u_0)} \end{aligned} \tag{D.4}$$

This process is repeated to the desired degree of accuracy.