1994

Automatic constraint-based synthesis of non-uniform rational B-spline surfaces

Philip Chacko Theruvakattil
Iowa State University

Order Number 9503600

Automatic constraint-based synthesis of non-uniform rational B-spline surfaces

Theruvakattil, Philip Chacko, Ph.D.

Iowa State University, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Automatic constraint-based synthesis of non-uniform rational B-spline surfaces

by

Philip Chacko Theruvakattil

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Mechanical Engineering
Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1994

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The road up to this point has be rather long and on occasions arduous, yet for the most part a thoroughly enjoyable and stimulating experience. Several people are responsible for making my experiences at school and college memorable and worthwhile. First and foremost, I wish to thank my parents for instilling in me the importance of education, a humanistic outlook on life, and for encouraging me to pursue graduate studies. My sincere gratitude to my wife, Lisa, for her support and encouragement over the past few years. How could I not acknowledge my brother, James, for sparring with me both intellectually and physically, thus helping me to grow in both mind and body during my formative years.

I wish to express my sincere gratitude to Dr. James Oliver and Dr. James Bernard for all their help and advice over the past few years. I would also like to thank Drs. Pierson, Rudolphi, Vanderploeg and Chen for graciously agreeing to be members on my committee. Thanks to Sriram Kini for helping me with programming and Kevin Renze for research related discussions. For miscellaneous computer related assistance, thanks to Jawad Mokhtar, Bob Kuehne, Jim Troy, Jim Schlosser, Pat Bergan and Jim Howard.

For all the pleasant diversions from the stress of academics and for helping to make my stay in Ames an enjoyable one, thanks to Omana, Jose, Christoph, Mathew, Nevin, Shawn, Rebecca, Lori and Balsy.

ABSTRACT

In this dissertation a technique for the synthesis of sculptured surface models subject to several constraints based on design and manufacturability requirements is presented. A design environment is specified as a collection of polyhedral models which represent components in the vicinity of the surface to be designed, or regions which the surface should avoid. Non-uniform rational B-splines (NURBS) are used for surface representation, and the control point locations are the design variables. For some problems the NURBS surface knots and/or weights are included as additional design variables. The primary functional constraint is a proximity metric which induces the surface to avoid a tolerance envelope around each component. Other functional constraints include: an area/arc-length constraint to counteract the expansion effect of the proximity constraint, orthogonality and parametric flow constraints (to maintain consistent surface topology and improve machinability of the surface), and local constraints on surface derivatives to exploit part symmetry. In addition, constraints based on surface curvatures may be incorporated to enhance machinability and induce the synthesis of developable surfaces.

The surface synthesis problem is formulated as an optimization problem. Traditional optimization techniques such as quasi-Newton, Nelder-Mead simplex and conjugate gradient, yield only "locally" good surface models. Consequently, simulated annealing (SA), a global optimization technique is implemented. SA successfully synthesizes several highly multimodal surface models where the traditional optimization methods failed. Results indicate that this technique has potential applications as a conceptual design tool supporting concurrent product and process development methods.

CHAPTER 1. INTRODUCTION

1.1. Overview

The reduction of product and process development time is a crucial factor for maintaining or improving overall competitiveness in manufacturing industries. Therefore, many companies are striving to reduce the number of physical prototypes and design iterations associated with the traditional design process. For such changes to occur, it is necessary to incorporate design and manufacturing process constraints as early as possible in the evolving product model. It is evident that in order to realize truly significant productivity enhancement, design tools as well as manufacturing processes must be improved in parallel.

In the past, the development of new design tools was often a consequence of a change in the design process. However, more recently, it appears that the necessary changes in the design process are defining the requirements and driving the development of new design technologies. For example, the recent development of feature-based modeling technology can be viewed as a manifestation of the growing emphasis on the philosophy of concurrent product and process design [Nevins and Whitney, 1989] which requires a close association between design geometry and production processes to ultimately reduce both physical prototyping and design changes late in the development process. The philosophy of concurrent product and process development is being actively researched and will undoubtedly influence design practice and methodology for some time to come.

The influencing forces driving solid geometric modeling toward feature based paradigms are also impacting one of the oldest "computer-aided" design tools, namely, sculptured surface modeling. Sculptured surface modeling technology is used extensively in the develop-

ment of a wide variety of consumer products, and industrial products such as car bodies, aircraft fuselages and ship hulls. Although this technology is used extensively, the methodology for creating surface models has changed very little over the past thirty years [Bohm et al., 1984; Farin, 1993; Faux and Pratt, 1979]. Traditionally, surface models are generated by various interpolation techniques, oftentimes based on the coordinate data sampled from a physical model. Consequently, the quality of the surface representation is dependent on the information content of the physical model at the time of data acquisition. Physical models are very expensive and it is very difficult to accurately incorporate changes and refinements of the physical model into the corresponding mathematical representation without completely recreating the mathematical model. This leads to a sequential design process which inhibits the introduction of simultaneous engineering practices.

1.1.1. An example

A typical example of changing design needs is provided by the automotive industry, in which the traditional development process has been serial in nature, driven to a large extent by the body stylist. A design starts with a set of theme sketches which are progressively refined and modified to generate an initial small-scale physical model. Production decisions are often made on the basis of these early sketches and models which may only incorporate very rough dimensional estimates of the proposed design. Development proceeds with the creation of a full scale clay model of the proposed design. This is typically a working model which is continuously refined based on engineering, manufacturing, and aesthetic criteria. For instance, coordinate measuring machines (CMM) are used to acquire cross-sections of point coordinate data to be used for preliminary packaging verification or various other rough engineering analyses.

As the design progresses and more detailed analyses are required, full scale wooden or plastic models ("bucks") of entire portions of the vehicle are developed based on the (clay)

model of the outer body. Ideally, the final shape of the vehicle converges when the clay model has been validated with respect to the various design criteria (e.g., engineering, manufacturing, marketing, etc.) and it satisfies a subjective aesthetic review. At this point no further changes to the clay model are allowed; it is digitized and modeled with sculptured surfaces. This mathematical model then becomes the basis for all further detailed analyses and tooling development.

Unfortunately, this process rarely proceeds according to plan. Often, during detailed analysis or testing, design deficiencies are discovered which necessitate changes that affect the outer body. Changes due to manufacturability constraints are also common. Somewhat less frequently, aesthetic, marketing, management, or other factors will change the design after model release. Outer body changes are first incorporated into the clay model, then the surface model is updated and re-released. Since the body is typically a smooth shape, changes in one area may affect several adjacent panels. Such model changes are time consuming and often have a ripple effect, causing substantial re-analysis and re-design. Furthermore, the later in the development process a change occurs, the greater its expense.

1.1.2. Methods for improvement

In response to increased competition, the automotive industry is striving to fundamentally reverse the current serial process of developing the outside of the vehicle first, then attempting to accommodate all structural, mechanical and occupancy requirements. This means future vehicle development will be much more parallel in nature wherein the interior components, aesthetics, and manufacturing process constraints influence the exterior shape during the conceptual design phase. This change reflects the trend toward incorporating more third-party and "carry-over" components in automotive design. However, this revolutionary change in the development process places new demands on body designers who must not only meet aesthetic design requirements, but also must simultaneously guarantee that a variety of

other design criteria are satisfied. This means sculptured surface models are required to capture more design information content.

Though the automotive industry is used as an example, such problems and challenges are not limited to automotive design. Similar difficulties are encountered in other design environments where sculptured surface modeling techniques are employed. For example, the aerospace and ship-building industries rely heavily on surfacing technology and typically incur large expenses in the design process. In the consumer products industries, surface modeling is used extensively in the design of a diverse variety of products, from telephones to tape dispensers. Like the automotive industry, many of these companies are under intense pressure to streamline the product development cycle to remain competitive. Given the wide spectrum of applications and the enormous potential payoff, the development of new strategies for surface synthesis is imperative.

1.2. Motivation and research objective

1.2.1. Motivation

The technology for modification of existing sculptured surface models is well understood [Cohen et al., 1980; Piegl, 1989], as is the link between such models and automated manufacturing process planning [Kim and Biegel, 1988; Wysocki et al., 1989]. Furthermore, recent advances in solid geometric modeling is leading to the robust incorporation (i.e., representation) of sculptured surfaces as an integral part of solid geometry [Casale and Bobrow, 1989; Saia et al., 1989]. Unfortunately, relatively few methods exist for the automatic generation of surface models subject to constraints derived from spatial, aesthetic, or other design and manufacturability requirements.

1.2.2. Research objective

The overall objective of this research is to study and develop methods for the automatic creation of sculptured surface models utilizing and incorporating design and manufacturing process information which typically exists (or can be derived) during the conceptual stage of product development. In this regard, the surface design problem is formulated as a optimization problem which will result in a shape that satisfies the various constraints imposed on it. Polyhedral models are used to represent interior components which dictate the overall spatial configuration of the surface. A distance metric which relates the surface to the polyhedral models is developed to evaluate the spatial satisfaction of the surface. Constraints on surface shape originating from aesthetic, functional and manufacturability requirements are accommodated by characterizing their relationship to analytic surface properties. The penalties due to constraint dissatisfaction are summed, resulting in a cost function which is subjected to an optimization procedure. In this case simulated annealing is used to arrive at the "best" design.

In contrast to traditional methods of generating a surface by interpolating a grid of selected points, the designer works instead with solid models of known or envisioned components to define obstacles which the surface must avoid by specified spatial tolerances. Additional input may include boundary or character curves and specified bounds on intrinsic surface properties generated from design constraints on functionality and manufacturability. Figure 1.1 depicts such a scenario for surface synthesis; a two-dimensional view of an automotive hood design with several constraints. This new technique will form the basis of a completely automated system for the synthesis of sculptured surface models. Such a tool is not envisioned to provide a highly precise, production-ready surface model, but rather an initial model that globally satisfies all specified constraints. This would provide surface designers with a rapid analytical prototyping facility, which would foster experimentation with novel configurations and enhance creativity.

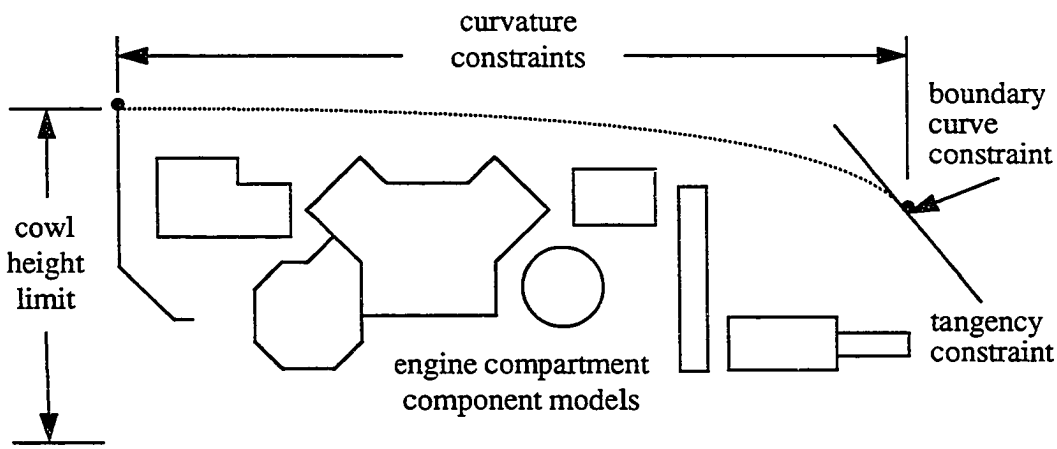


Figure 1.1 Surface design synthesis (surface in dashed line)

Various mathematical representations of curves and surfaces are briefly discussed in the following chapter. An introduction to current surface design methodology is also presented, which includes a discussion of traditional surface modeling techniques and recent research in automated surface smoothing. A brief description of energy based surface generation and particle based surface modeling techniques is also included. In the third chapter, the general sculptured surface synthesis problem is developed along with the various constraints that may be applied, which includes both geometric and intrinsic shape constraints. A description of the distance evaluation algorithm is included in Chapter 3. The surface synthesis problem is posed as an optimization problem. In Chapter 4, five different optimization algorithms are discussed: Nelder-Mead simplex, conjugate gradient, quasi-Newton, and two versions of simulated annealing. Examples using all optimization techniques are presented in Chapter 5. Based on the results of the experiments in Chapter 5, one version of simulated annealing is chosen for surface synthesis. In Chapter 6 some additional examples are presented. Finally, in the last chapter, conclusions are drawn and some possibilities for future research are suggested.

CHAPTER 2. CURVES AND SURFACES

2.1. Curve and surface representations

2.1.1. Introduction

In the days before the advent of the computer, surfaces were defined by a set of curves, usually planar sections, plus some characteristic feature lines [Bohm et. al., 1984]. Once this information is available, templates can be manufactured and the templates can in turn be used to produce master models. The master models may then used to make the final stamps and dies by means of copymilling. However, this technique does not lend a convenient methodology for the stylist who thinks in terms of "character lines" which are seldom plane curves [Farin, 1993]. Hence the need for defining and manipulating "free-form" curves and surfaces. However, it was not until the late fifties that research towards mathematical representations started to take root. This was driven, to a certain extent, by the need to store the surface definition in a computer-compatible format so as to transfer this information directly to milling machines driven by "numerical control" [Bohm et al., 1984]. Most of the early research on curves and surfaces was funded by the automobile and aircraft industry [Watt and Watt, 1992]. This was probably due to the need for high quality surface definition so as to attract customers.

One of the earlier techniques for generating computer models was to digitize the actual physical model and then develop a mathematical model using one of several methods i.e. interpolation, approximation or lofting. This technique is useful when a product has to undergo design changes and a computer model is not available, which was often the case in the early days of computer-aided design. It is also possible that competing manufacturers of

products make use of such techniques to create mathematical models which can then be subject to further analysis/modification. It was also recognized that conceptual design could be carried out on a computer by manipulating free-form curves and surfaces.

Another field where curve and surface fitting finds considerable use is, the natural sciences where experimental data is to be fit with a curve or surface. However, these techniques do not constitute design per se. In general, a CAD package must provide techniques for interpolating and modifying data points obtained from a digitized model, allow for interactive input and free-form design. It should also be able to represent standard analytic shapes such as lines, conics, circles, planes, and quadric surfaces precisely. The first part of this chapter enumerates the various mathematical representations of curves and surfaces. Arguments for and against each representation scheme are included. The representation schemes described are by no means exhaustive. However, the representation schemes chosen for inclusion in this dissertation follow a rather "loose" chronological order.

2.1.2. The nonparametric form

Curves and surfaces may be represented nonparametrically or parametrically. Nonparametric forms may be explicit $y = f(x)$ (curve) or implicit $f(x, y) = 0$ (curve). The explicit form is a one-to-one relationship thus, it cannot be used to represent closed or multivalued curves or surfaces. Some of the other limitations of both the explicit and implicit nonparametric representations are:

- The tangent (tangent plane) at a point on the curve (surface) may be vertical or near vertical and that leads to values (infinity or very large) that cannot be handled computationally.
- In general, the interest is in the object and not it's relationship to some coordinate system. However, the choice of a nonparametric representation imposes a relationship between the object and coordinate system.

- Often the description of shape of an object requires very complex curves/surfaces which cannot be represented by simple nonparametric functions or even by single parametric curves/surfaces. However, the parametric representation provides for a convenient piecewise description (composite curves and surfaces) [Watt and Watt, 1992].

The main advantage of the nonparametric form is the possibility of quickly determining whether a point lies on the curve/surface or on which side of the curve/surface the point lies. In addition, normals are easily computed [Foley et al., 1992]. However, the problems presented by the nonparametric form motivated the use of the parametric form of curve and surface representation which does not pose the same problems. There are several flavors of parametric curve and surface representation. Some of the more popular parametric forms used in CAD are described in the following section.

2.1.3. The parametric representation

2.1.3.1. The monomial and Lagrange form Parametric forms of curves and surfaces overcome the problems posed by their nonparametric counterparts. The parametric representation of a space curve is: $x = x(u), y = y(u), z = z(u)$ where u is the parameter. The parametric representation of a surface is: $x = x(u, v), y = y(u, v), z = z(u, v)$ where u and v are parameters. Figure 2.1 shows the relationship between the parameter space and the object space. Usually the range of the parameter space is $[0, 1]$. Given the problems associated with nonparametric representations, parametric representations are currently the choice for computer-aided modeling of curves and surfaces.

The most common parametric forms of curves and surfaces and are polynomial and rational polynomial. Other forms such as Fourier series are possible but not used in CAD or graphics applications because the number of computations involved are greater than with polynomials. The monomial form of a polynomial curve of degree k or order $k+1$ is:

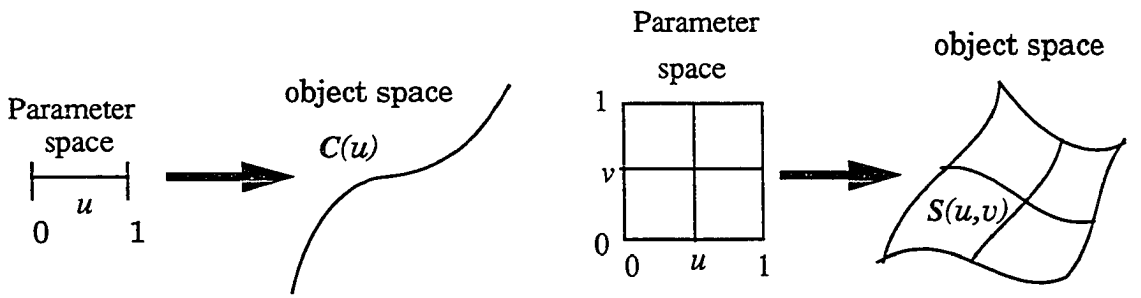


Figure 2.1 Relationship between parameter space and actual

$$\boldsymbol{P}(u) = a_0 + a_1 u + a_2 u^2 + \ldots + a_n u^n \tag{2.1}$$

where u is the local parameter. Cubic polynomials are the most commonly used, for design applications because lower degree polynomials do not offer sufficient shape flexibility [Foley et al., 1992]. In addition, cubics are the lowest degree curves that are nonplanar in 3D and provide C^1 and C^2 continuity. A polynomial of degree n may have up to $n-1$ turning points implying that the greater n, the more oscillatory the curve. This is a problem in some situations when using the Bezier form (described later in this section). In some applications higher degree curves and surfaces are used because higher degree derivatives are required to control the shapes of the curves and surfaces that are being designed. Curvature information may be used for better parametrization of curves or surfaces so as to achieve a good compromise between speed and quality of rendering [Kosters, 1991].

Regardless of the degree of the monomial representation, the relationship between the shape of the curve and the coefficients is not intuitive or clear, except at the point $u = 0$. This makes it difficult for a designer to specify and interact with the curve shape using these coefficients. The same can be said for surfaces. Finally, the monomial form is not affine invariant which almost immediately rules it out for CAD or graphics purposes. Affine combinations are weighted sums of points where the weights sum to one:

$$b = \sum_{j=0}^{n} \alpha_j b_j \qquad \begin{cases} b_j \in \Re^3 & \text{points} \\ \alpha_0 + \alpha_1 + \ldots + \alpha_n = 1 & \text{weights} \end{cases} \tag{2.2}$$

Convex combinations are an important special case of affine combinatons wherein the coefficients, in addition to summing to one, are also nonnegative. Transformations used in computer graphics and CAD such as translation, scaling, shear, rotation, and parallel projection are all affine maps. An affine map is one that leaves affine combinations invariant. [Farin, 1993]

One may use Lagrangian interpolation to overcome these problems. The following is a curve representation using Lagrangian interpolation,

$$\underline{p}(u) = \sum_{i=0}^{n} \underline{p}_i L_i^n(u) \tag{2.3}$$

where the L_i^n are the Lagrange polynomials,

$$L_i^n(u) = \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (u - u_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (u_i - u_j)} \tag{2.4}$$

Lagrange polynomials suffer the same deficiency as the monomial representation for large values of n. Furthermore, the monomial and Lagrangian forms do not lend themselves to the discussion of smoothness properties of piecewise curves [Bohm, et al., 1984].

2.1.3.2. Hermite form Hermite polynomials produce curves that interpolate two endpoints and the derivatives at the endpoints. In the case of a bicubic Hermite surface the four corner points, the derivatives in the two parameter directions and the twist vectors at the corners are used to determine the necessary sixteen coefficients. This technique provides a partially geometrically intuitive method wherein the endpoints and/or endpoint derivatives may be used as shape handles, however, the twists must generally be estimated which is not very intuitive. A number of techniques for estimating the twist are discussed by Farin [1993].

Also, any change of endpoint or derivative influences the entire curve or surface segment. Another disadvantage of this method of interpolation is that it is not invariant under affine transformations of the parameter domain. That is, given an initial parameter domain of $[0, 1]$ which has to be transformed to $[a, b]$, unless the tangent vectors are manipulated the resulting curve will be different [Farin, 1993].

2.1.3.3. The Bezier form The Bezier form of curve/surface representation has been very popular and widely used for computer-aided design. This form overcomes most of the problems encountered by the representations discussed earlier. In addition the control polygon/net of a Bezier curve/surface provides the designer with a geometric sense of the shape of the curve/surface. The Bezier curve representation in terms of Bernstein polynomials may be written as:

$$b(u) = \sum_{i=0}^{n} b_i B_i^n(u) \tag{2.5}$$

where the Bernstein polynomials are defined as:

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} \tag{2.6}$$

Some properties of the Bezier form are:

- Affine invariance. An important property because curves and surfaces in a CAD system need to be repositioned, scaled etc. Affine invariance allows an affine map to be applied to the control polygon before the curve is evaluated.
- Invariance under affine parameter domain transformations. A property not available with Hermite curves.
- Convex hull property. The curve lies within the convex hull of the control polygon. A useful property for interference checking [Farin, 1993].
- Endpoint interpolation. The Bezier curve interpolates (passes through) the first and

the last control points. This is useful in many design situations.

- The Bernstein polynomials are symmetric with respect to u and $1 - u$.

Bezier curves have some other useful properties which are discussed by Farin [1993].

The main disadvantage of the Bezier representation is that any design change affects the entire curve/surface. This problem can be handled by using composite Bezier curves. However, there may be some difficulty associated with connecting individual curve/surface segments depending on continuity constraints to be applied at segment joints [Watt and Watt, 1992]. Bezier surfaces have analogous properties to and advantages of the Bezier curve. For a detailed discussion of Bezier curves and surfaces see Farin [1993] or Faux and Pratt [1979].

2.1.3.4. B-spline representation The B-spline representation is a generalization of the Bezier representation and was first introduced into CAD by Riesenfeld [1973]. B-splines are completely defined by the degree of the curve/surface, the knot vector sequence, and the control polygon/net. For a detailed discussion of B-splines see de Boor [1972], Farin [1993], Faux and Pratt [1979]. The B-spline representation shares most of the characteristics of the Bezier representation. In addition, B-splines allow for local shape control, and continuity between curve/surface segments are inherently satisfied. Control points may be added without increasing or decreasing the degree of the curve/surface which is a very useful feature. B-splines may be parametrized uniformly or nonuniformly. Uniform parametrization is used when the global parameter values occur at equal intervals and nonuniform parametrization is used when the global parameter is unevenly spaced. B-splines may be used for interpolation or for representation of free-form curves and surfaces. Furthermore, the use of nonuniform parametrization provides additional degrees of freedom for controlling the shape of the curve/surface, enabling a larger class of shapes.

For CAD applications, analytic shapes such as lines, circles, conics and quadrics are important. Unfortunately, non-rational B-splines cannot precisely represent standard analytic shapes. The fact that non-rational curves are in general not invariant under perspective trans-

formations could lead to expensive computations. This led to NURBS (nonuniform rational B-splines), a unified mathematical form for the precise representation of free-form curves and surfaces as well as standard analytic shapes that remain invariant under perspective transformations.

2.1.3.5. NURBS Rational curves were first introduced to CAD by Coons [Farin, 1993]. Homogeneous coordinates are used to represent a rational curve in n-dimensional space as a polynomial in $(n+1)$-dimensional space. A NURBS curve is defined by a set of four dimensional control points: $p_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$ (where the w's are weights) and knot vector $U = \{u_0, \ldots, u_{n+p}\}$ a monotonically increasing set of real numbers. The perspective map in 3D of such a curve is called a rational B-spline curve [Piegl and Tiller, 1987]:

$$\boldsymbol{C}(u) = H\{\boldsymbol{C}^w(u)\} = H\{\sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P}_i^w\} \tag{2.7}$$

$$= \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \boldsymbol{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} = \sum_{i=0}^{n} R_{i,p}(u)\,(H\{\boldsymbol{P}_i^w\}) \tag{2.8}$$

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{2.9}$$

Likewise the perspective map in 3-D of a tensor product surface is called a rational B-spline surface of degree (p, q)

$$\boldsymbol{S}(u,v) = H\{\boldsymbol{S}^w(u,v)\} = H\{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v)\boldsymbol{P}_{ij}^w\} \tag{2.10}$$

$$= \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{ij}\boldsymbol{P}_{ij}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{ij}} = \sum_{i=0}^{m}\sum_{j=0}^{n} R_{i,p;j,q}(u,v)\ (H\{\boldsymbol{P}_i^w\}) \qquad (2.11)$$

where

$$R_{i,p;j,q}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{ij}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{ij}} \qquad (2.12)$$

The normalization of a perspective map is defined as,

$$H\{(wx, wy, wz, w)\} = \begin{cases} (\frac{wx}{w}, \frac{wy}{w}, \frac{wz}{w}) & \text{if } w \neq 0 \\ \text{points at infinity on the line from the origin through (x,y,z)} & \text{if } w=0 \end{cases} \qquad (2.13)$$

The $N_{i,k}(u)$ are the kth degree B-spline basis functions which are recursively defined following the Cox-de Boor algorithm [Cox, 1972; de Boor, 1972]:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u \le u_{i+1} \text{ and } u_i < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad (2.14)$$

$$N_{i,k}(u) = \frac{u-u_i}{u_{i+k}-u_i}N_{i,k-1}(u) + \frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}}N_{i+1,k-1}(u)$$

$\boldsymbol{P}_i$ and $\boldsymbol{P}_{ij}$ are the control points of the curve and surface respectively. When evaluating the basis functions it is assumed that $0/0 = 0$.

The blending functions are defined over the entire real line but have influence only on the interval $u \in [u_0, u_m]$. The u_i are the knots forming a knot vector $U = \{u_0, \ldots, u_m\}$. The knot vector governs the relationship between parametric and spatial variation, and its entries represent the parameter values at the segment joints (knots). The number of knots,

degree and the number of control points are related by the formula $m = n + k + 1$. When the end knots are repeated with multiplicity $k + 1$ the resulting B-splines are called nonperiodic and in this case the end control points are interpolated.

In the case of surfaces, knot vectors are required in both parameter directions (u, v) and similar relationships exist between the number of knots, the degree and number of control points. The w_i and w_{ij} are the weights for rational B-spline curves and surfaces respectively. The weights provide the flexibility to design a larger variety of shapes. in each parameter direction. A NURBS surface is in general not a tensor product surface [Bohm et al., 1984].

In recent years, NURBS have emerged as a defacto standard for surface modeling applications due to its many attractive attributes. Consequently, NURBS are the choice for curve and surface representation in all the research documented in this dissertation. The above description of NURBS is only a summary to justify its use in this research. For a detailed discussion of NURBS and their attributes see [Tiller, 1983; Piegl and Tiller, 1987; Piegl, 1991; Farin, 1993].

2.2. Design synthesis

Design of a product may be primarily evolutionary in nature or it may be the result of innovation. Evolutionary design depends on existing information hence, it is conducive to computerization from inception. In the event of evolutionary design, a model of the product is either already available as a computer model or it has to be transferred to the computer. In the latter case the physical model is digitized, using lasers or CMM's for example, after which a computer model is generated by interpolation, approximation, lofting etc. If the capability to digitize a physical model does not exist or if the model is simple enough, then it may be modeled interactively. Interactive generation of computer models can be quite tedious and time consuming depending on the complexity of the model and the functionality of the CAD tool being used. Once a computer model of a product is available, it can be analyzed (finite ele-

ment analysis for example) and then redesigned based on the results of the analysis. Thus improvements in the design of a product or an entirely new product results. In the automobile industry, a significant number of new designs are still a product of this evolutionary design process [Tovey, 1989].

In the early phase of innovative design, computer involvement is limited because of the large number of unknowns. Recently, Jensen et al., [1991] reported the development of curve and surface algorithms to minimize geometric constraints. They expect such curves and surfaces to be useful for the specification of shape via sketching. This tool could be used by stylists involved in innovative design of products. Despite attempts at computerizing the designs of stylists, most current CAD packages are not good enough. The main reason is that most CAD packages are subjected to certain constraints and require the user to follow certain procedures, both of which are restrictive on the creativity of the stylist. Oftentimes, even in the process of innovative design a computer model is interactively generated from the theme sketches of the stylist after it has undergone several revisions and has met the approval of management. These designs are then analyzed or their operation is simulated to either modify or eliminate the various candidates. In mathematical terms, more often than not, this amounts to providing a solution to a nonlinear, constrained optimization problem with many variables and constraints. This apparent inefficiency has spurred research activity in the automated synthesis of component shape. The goal is to relieve the designer of the task of manually creating design candidates by automatically generating configurations which are formulated on the basis of the specified spatial and design constraints. Design synthesis research to date has primarily concentrated on the structural aspects of mechanical elements, e.g., trusses, frames, brackets, etc. From the standpoint of automated surface synthesis, these techniques are not directly extensible, although certain concepts may be applicable.

One approach to the shape synthesis problem is shape optimization, which is essentially a combination of optimization and finite element analysis, and more recently, boundary

element techniques. Shape optimization problems can be categorized as: sizing, and geometrical and topological optimization [Haftka and Gurdal, 1993]. In the case of sizing optimization one determines, for example, the optimal thickness distribution for an elastic plate structure subjected to given boundary and loading conditions. Geometric shape optimization allows for boundary movement by introducing additional design variables. Topology optimization involves the determination of features such as the number of holes in the domain and the connectivity of the nodes. Shape optimization methods have been studied extensively [Bennett and Botkin, 1986; Haftka and Gurdal, 1993], and there continues to be much research in this area. A disadvantage of the sizing and geometric shape optimization techniques is that the choice of structural topology, which must be specified at the outset, severely constrains the solution. Imam [1982], suggests an adaptive re-meshing scheme to address this problem, while Shah [1988], presents a combined algorithmic and heuristic technique to generate good candidates for initial form. The common feature of most shape optimization techniques is the iterative application of finite element analysis. However, this approach is practical only for simple geometries.

The shape optimization problem can be related to the surface synthesis problem. Williams [1987], uses a structural analogy between curved shell finite elements and parametric surface patches to formulate a technique for smoothing surfaces by minimizing strain energy. However, this method is computationally expensive since it involves the solution of nonlinear differential equations. Also, like the other shape optimization techniques this approach does not synthesize geometry, but rather modifies it in some local sense. Within this framework, there is no clear way to incorporate global aspects of the surface synthesis problem, such as spatial constraints.

The application of knowledge-based system technology to the design synthesis problem is one of the most active areas of design research. In a manner analogous to the shape optimization techniques, early efforts [Dixon et al., 1984] focused on the generation of accept-

able design parameters given a general configuration and pertinent domain knowledge. More recently the focus has shifted to automating the generation of structural configuration (topology) as well as parametric dimensions. Welch and Dixon [1989], describe a system for the design of sheet metal parts with separate but interacting modules for configuration design and parameter generation. The configuration level allows design storage and backtracking, and both levels employ iterative redesign, so that a best-first search of the design space is accomplished. Related techniques that deal with the topological synthesis of 2-D frame structures have also been developed [Nevill and Paul, 1987]. Weitzman [1992] developed an interactive tool for assisting with the design of two-dimensional graphic interfaces for instructional systems.

However, these systems are limited in their domain by the extent of the knowledge base. Generally knowledge-based solutions do not scale up well, i.e., techniques that work well in a relatively small, well defined domain, but may not be feasible in a much larger domain such as the surface synthesis problem. Such an approach, however, may be useful for solving specific sub-problems within the larger framework. A knowledge-based approach may well be suited for finding the initial topological distribution of surface patches: a logical collection of triangular and rectangular patch topologies, for instance.

2.3. Surface design

With roots in both the automotive and aircraft industries, sculptured surface modeling techniques emerged in the early 1960's, in response to the growing capabilities of numerically controlled machining technology and the subsequent need to represent smooth surfaces numerically. Some schemes for curve and surface representation along with advantages and disadvantages of each scheme was presented earlier in this chapter. Surface modeling may be separated into two broad categories: traditional surface modeling and automated surface modeling. Automated surface modeling techniques are based on standard surface representations,

on a physically-based modeling scheme or on particle based schemes. In the following sections, these surface modeling techniques are discussed.

2.3.1. Traditional surface modeling

Traditional surface modeling techniques consist of interactive control point manipulation, interpolation and approximation of surface points, lofting schemes, boolean set operations and other specialized schemes [Cobb, 1984]. Of the techniques mentioned above, the most popular have been interactive control point manipulation, interpolation of data points and lofting schemes.

Interpolation techniques were motivated by the need to translate a physical model or drawing into a mathematical representation. This process typically involves the discretization of the object into a large set of point coordinates. The data points may be obtained by digitizing an already existing model or by interactive specification, which could be very time consuming. Various techniques exist to interpolate such data to produce smoothly varying bivariate surface functions.

Interactive control point manipulation allows for easier and quicker "free-form" surface design, i.e., the conceptualizing surfaces completely analytically, without initial physical models.

These two design philosophies form the basis for a general categorization of surface modeling techniques as either transfinite methods (typically associated with S. A. Coons) or tensor product methods (attributed to P. Bezier). Although these techniques were developed contemporaneously, and either one can be applied to both design paradigms, transfinite methods are most often applied to generate surfaces from physical models, while tensor product methods are generally used for free-form surface styling [Bohm et al., 1984].

The transfinite surface representation builds on the idea of a ruled surface which, in its simplest form, is a linear interpolation of two space curves which have the same parametriza-

tion. Given four boundary curves $p_0(u)$, $p_1(u)$, $q_0(v)$, $q_0(v)$, Coons observed that a smoothly varying surface between them could be formulated as the "Boolean sum" of the two ruled surfaces, $\boldsymbol{P}(u, v)$ and $\boldsymbol{Q}(u, v)$, formed from opposite boundary curves. Thus, a Coons surface (patch) is defined as,

$$\boldsymbol{S}(u, v) = \boldsymbol{P} \oplus \boldsymbol{Q} = \boldsymbol{P}(u, v) + \boldsymbol{Q}(u, v) - \boldsymbol{PQ}(u, v) \tag{2.15}$$

where $\boldsymbol{PQ}(u, v)$ is a doubly ruled surface which simply blends the four curve end-points. The piecewise implementation of this technique often led to problems in maintaining continuity across adjacent patches. Subsequently, the Gordon surface was developed as a generalization of the Coons patch over a network of intersecting curves. A typical modeling procedure (for a topologically rectangular surface) involves, as a first step, the interpolation of curves in both parametric directions through a regular grid of digitized data points. Due to variations in the data acquisition and interpolation techniques the parametrically orthogonal curves rarely intersect in Euclidean space. Thus the most tedious part of the modeling task is to manipulate the curves until they mutually intersect to within machine precision. When this is accomplished, the model is complete, since a Gordon surface is completely defined by its bounding curves.

The tensor product surface can be viewed as a generalization of curve interpolation techniques. Although they can be described in terms of any interpolative basis, the Bernstein basis of Bezier (and more generally B-spline) curves is used here due to its intuitive nature. Consider a cubic Bezier curve in the plane of the paper defined by four control points $\boldsymbol{b}_i$ and the function,

$$\boldsymbol{b}(u) = \sum_{i=0}^{3} \boldsymbol{b}_i B_i(u) \qquad u \in [0, 1] \tag{2.16}$$

where $B_i(u)$ are the cubic Bezier blending functions. If this curve is swept normal to the paper and each control point is allowed to trace out a Bezier curve $\boldsymbol{b}_i(v)$ such that,

$$b_i(v) = \sum_{j=0}^{3} c_{i,j} B_j(v) \qquad v \in [0, 1] \tag{2.17}$$

then the resulting tensor product (Bezier) surface is defined by,

$$T(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} c_{ij} B_i(u) B_j(v) \qquad u, v \in [0, 1] \tag{2.18}$$

This technique of surface design is typically implemented by allowing the user to interactively create a series of approximately parallel curves. These could be created by simply allowing the user to select and modify control points, or the system may initially interpolate curves (i.e., calculate the governing control points) which pass through a series of data points. Assuming that the series of roughly parallel curves has the same parametrization, the tensor product surface is obtained by interpolating the corresponding control points from each curve in the opposite parametric direction. In the piecewise construction of tensor product surfaces, the user must take special measures to ensure derivative continuity. Thus with a relationship analogous to that between Gordon and Coons surfaces, the B-spline surface evolved as a generalization of the Bezier surface patch and the NURBS surface evolved as a generalization of a B-spline surface.

2.3.2. Automated surface generation

Much of the research toward automated surface generation has followed a procedure analogous to the shape optimization approach, i.e., given an initial model which meets most design constraints, generate a slight perturbation in order to optimize it with respect to some other constraint, typically curvature. For example, Andersson et al., [1988] minimize control point change with a linearized distance metric, subject to linearized normal curvature constraints, to ensure convexity. An iterative solution is proposed in which the surface is sampled to find a set of points which violate the constraint, then the linear programming problem is

solved with Karmarkar's algorithm [Karmarkar, 1984] and the process is repeated until a stable solution is reached. Lott and Pullin [1988], reverse this formulation by using a measure of surface curvature as the objective function, subject to a constraint based on the distance between the original and modified surfaces. Both the objective and constraint functions are linearized in the surface parameter space and the simplex method [Reklaitis et al., 1983] is used to minimize the curvature metric. Both these approaches reduce the degrees of freedom of the problem by restricting the control points to deviate only along normals of the original surface. In contrast, Ferguson [1986] develops convexity conditions on the governing control point net to eliminate spurious inflections from B-spline curves. This approach is extended to surfaces [Ferguson et al., 1988] by projecting the control point net onto an user specified plane. General perturbations are allowed, and a full nonlinearly constrained optimization problem is formulated and solved with successive quadratic programming. It is interesting to note that these three approaches address a similar problem from three different applications: Andersson et al., [1988] deals with automotive design, Lott and Pullin [1988] with ship building and Ferguson et al., [1988] with aircraft surfaces.

A similar approach is surface smoothing based on reflection lines [Kaufmann and Klass, 1988; Poeschl, 1984]. These techniques simulate the reflection of lights on a reflective surface to assess its quality. Such methods are not as "finely tuned" as the curvature-based methods described above, since reflection lines are based on a first order interrogation, while curvature involves second order interrogation. However, they may be more useful in detecting global imperfections [Farin, 1993].

2.3.2.1. Physically based methods Terzopoulos [1987], and Platt and Barr [1988], present techniques for physical system simulation based on classical mechanics. Celniker and Gossard [1989, 1991], use a similar approach in the development of an energy based system for free-form surface design. This system allows the designer to control surface shape by imposing boundary conditions and external loads. The surface generated is not a sculptured

surface in the traditional sense but rather a wireframe which connects points that behave according to the physical model.

Current surface synthesis methods impose, on the designer, the task of ensuring that the surface not only accommodates the global environment in which it must function, but also all smoothness criteria derived from aesthetic, performance, manufacturability or other design requirements. Obviously, most of the effort toward automation has focused on the latter of these tasks. The energy-based technique, by providing control via external loading, comes the closest in spirit to global surface synthesis. But the resulting surface model is inadequate for the precise representation and processing required by many products. Furthermore, application of boundary conditions and external loads requires skills not necessarily possessed by designers.

2.3.2.2. Particle based modeling Initially, particle based paradigms were used to model natural random events such as fires and waterfalls. In such models, force fields and constraints influence the movement of particles but the particles do not interact with each other. The next stage in the development of particle based models incorporated particle interaction by introducing spherically symmetric potential fields around the particles. However, this naturally leads to volumes rather than surfaces. Recently, Szleliski and Tonnesen [1992], introduced the concept of oriented particles systems, wherein the particles are induced to form surfaces instead of solids. As in the case of physically based modeling, the designer has to deal with quantities that he/she may not be familiar with such as force fields.

2.4. Summary and conclusions

Some of the important curve and surface representations such as the Hermite form, the Coons representation, the Bezier form, the B-spline representation and NURBS were briefly described. For every representation discussed, the main advantages and disadvantages are listed. This research focuses on the generation of NURBS based surfaces because of its emer-

gence as the defacto standard for curve and surface representation and its many attractive attributes.

In Section 2.3, some of the traditional and contemporary surface modeling techniques are described. Of these techniques, physically based modeling comes closest in spirit to what is proposed in this dissertation. The need to use forces and boundary conditions as design handles is not necessarily intuitive to a designer. Particle based modeling is a very powerful technique, but suffers from the need to convert between such models and other representations on which most CAD packages are based.

CHAPTER 3. PROBLEM FORMULATION

Consider a simple surface synthesis problem posed as follows. A sculptured surface model $S(u, v)$ is desired in the vicinity of a single polyhedral model Q as shown in Figure 3.1. The polyhedral model could represent a car engine for example. The general spatial location of the surface $S(u, v)$ relative to the polyhedral model Q is known and is approximated initially by a planar surface as shown in Figure 3.1. To characterize the desired functional behavior of the

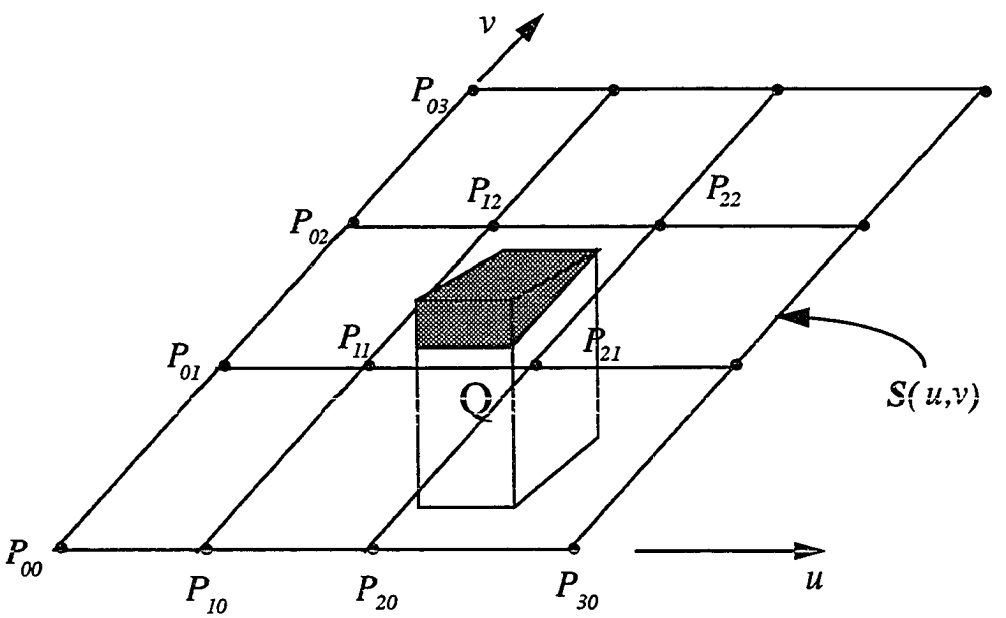


Figure 3.1 Surface synthesis, initial configuration

surface, a set of constraints is formulated to induce the surface to avoid the polyhedral model by at least some tolerance δ. These constraints tend to characterize the global behavior of the surface. In addition, a number of other design constraints, based on surface functionality and manufacturability, for instance, can be formulated in terms of analytical surface properties.

These constraints generally have a subtle local influence on the overall shape of the surface. To maintain the rectangular topology of the surface patch, shown in Figure 3.1, some of the control points (generally the corners) are designated as fixed. The coordinates of the remaining control points are the design variables of the surface synthesis problem, i.e., a surface is synthesized by finding suitable positions for the free control points such that all the constraints are satisfied. It should be noted that the knots and/or the weights can also be used as design variables.

For every constraint imposed on the surface, a cost is computed based on the level of constraint dissatisfaction. The cost increases with increasing constraint dissatisfaction. The cost components are then summed to yield the overall cost of the current surface configuration. The next logical step is to seek a new configuration that yields a lower overall cost. This could be achieved by perturbing the position of the control points and re-evaluating the total cost. This process would continue until a configuration with the lowest possible cost is achieved which is nothing but a process of configuration optimization. Considering that a given problem could have several variables, it would be very tedious to manipulate the surface manually to achieve a better configuration at every step. Furthermore, manual positioning of the control points may not lead to the best surface. Consequently, an optimization technique is used to obtain the surface configuration that best satisfies the constraints. A description of the optimization techniques used in this research is included in the following chapter.

Since the overall cost function for this problem must accommodate components derived from several diverse contributing factors, a reasonable implementation is to sample the surface at a number of points, possibly at regular parametric intervals. The cost contribution for dissatisfaction of some constraints is the sum of the cost incurred by each sample point. Alternatively, for some constraints, the cost is computed for each constant parameter curve and then summed. The individual cost components are defined as functions of surface point coordinates $S(u, v)$ and/or its derivatives. They also depend on the number of samples M and N

in the u and v parametric directions respectively. Thus M and N are used as input parameters having the expected trade-off effect between accuracy and convergence on the one hand and computational efficiency on the other. For most of the examples described in this dissertation the surfaces are sampled at regular parametric intervals, i.e.,

$$u_0 = 0$$
$$u_i = u_{i-1} + \frac{1}{M-1} \qquad i = 1, 2, \ldots, M-1 \tag{3.1}$$

$$v_0 = 0$$
$$v_i = v_{i-1} + \frac{1}{N-1} \qquad i = 1, 2, \ldots, N-1 \tag{3.2}$$

For a given configuration the total cost C_{total} is a sum of the cost incurred for dissatisfaction of the constraints C_i that are active,

$$C_{total} = \sum_{i=1}^{a} C_i \tag{3.3}$$

where a is the number of active constraints. The constraints developed during the course of this research are described in the following section. For the experiments conducted, various combinations of these constraints are imposed. The combination of constraints used for each example problem will be discussed with the results. The following section describes the various constraints developed during the course of this research.

3.1. Development of constraints

3.1.1. Proximity constraint

This constraint is a result of the requirement that the surface not interfere with the polyhedral models. This constraint results in a proximity cost component, which requires a function to determine the distance from an arbitrary point in three space to a polyhedral model. The

sophistication of such a distance function is dictated by the complexity of the polyhedral model. In this research only convex polyhedral models are considered. However, nonconvex models can be dealt with by decomposing them into a collection of convex models [Woo and Shin, 1985]. Nonconvex polyhedral models are part of the design environment in some of the experiments conducted.

Given a polyhedral model Q comprised of V vertices, a distance function $dist(S((u_i, v_j), Q))$ is formulated such that it returns the minimum distance from a surface point to the obstacle, D_{ij}. The distance function algorithm must be efficient, since it will generally be called many times. All polyhedral model information is stored in the BYU format while surface models are stored in the IGES format. BYU files consist of the vertex coordinates along with a connectivity list for each face such that the face normals are outward pointing. The BYU file along with the IGES files, are read at the beginning and processed. An outline of the preprocessing and the distance function procedures follows.

3.1.1.1. Preprocessing The preprocessing step is described in this section.

- Read BYU file and store the vertex information in a data structure created specifically for the polyhedral models. Use the vertex information to determine the edges of the polyhedral model and store this information in the data structure.
- Use vertex information and connectivities to compute face normals F_i^n for each face i of a polyhedral model. For each face determine the equations of the edges that make up each face E_{ij} and the equations of the planes which contain the edges and are normal to the corresponding face EP_{ij}. This information is then stored in the data structure which is accessed while computing the distance from a surface point to the polyhedral models.

3.1.1.2. Distance function The computations involved exploit the stationarity of the polyhedral models.

1) For a given surface point $S(u_i, v_j)$ determine if the point is within the bounding box plus tolerance (see Figure 3.2) of one or more polyhedral models. If the point is outside the bounding box of all the polyhedral models then there is no need to compute the distance because this means that the point satisfies the proximity constraint.

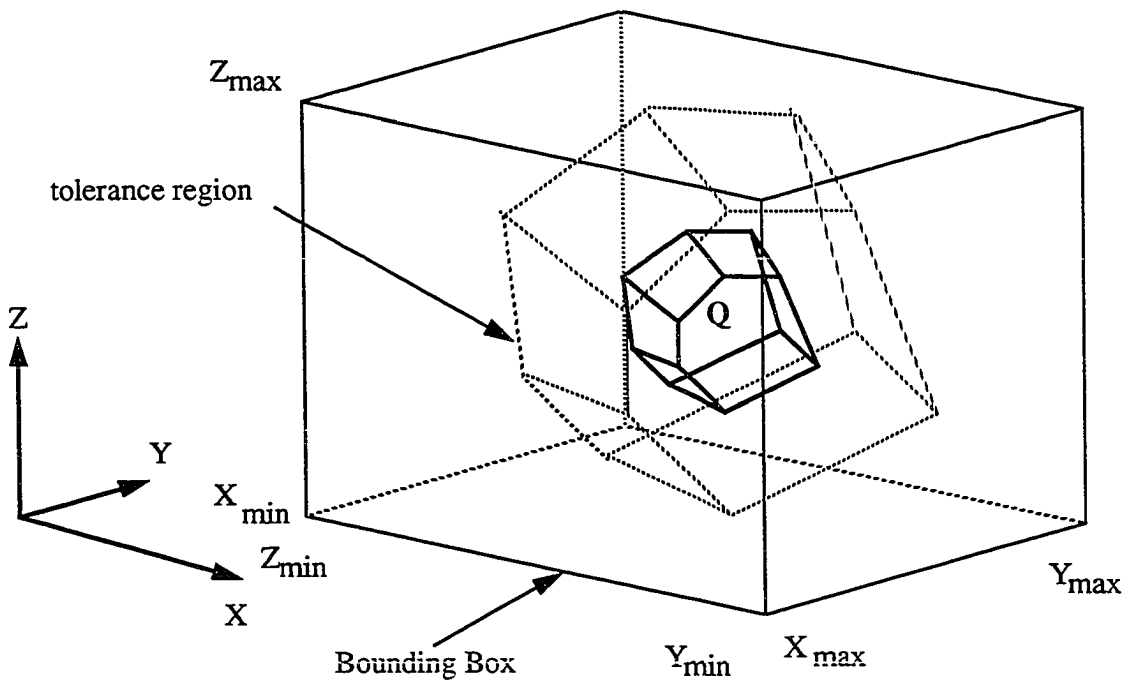


Figure 3.2 A polyhedral model with bounding box

2) If the point is within the bounding box of a polyhedral model then proceed to compute the distance. For the polyhedral model under consideration, compute the scalar product of the surface point with each of the face normals $S(u_i, v_j) \cdot F_k^n = D_k$. If all the D_k's are negative then the point is within the polyhedral model. Therefore, return a -1 to the calling function.

3) If even one of the D_k's is positive it implies that the point is outside of the polyhe-

dral model. Determine the face k that yields the largest positive D_k. The surface point $S(u_i, v_j)$ (S^1 or S^2 in Figure 3.3) is projected (S_p^1 or S_p^2 in Figure 3.3) on to the plane containing the face that yields the maximum positive D_k. A check to determine if the projected point S_p is inside the face or outside, is described in the next step.

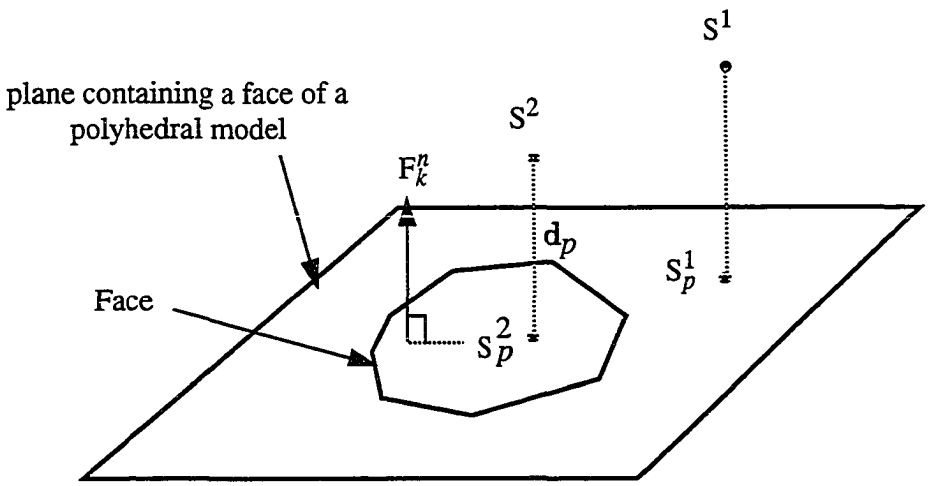


Figure 3.3 Projection of a point onto a plane

4) Starting at the first face edge find the scalar product of EP_{ij} and S_p^1 in this case DE_{ij}. Do the same for all the edges of the current face. If all DE_{ij}'s are negative then the projected point is within the face. If any of the DE_{ij}'s is positive then the projected point lies outside the current face. In Figure 3.3, the projected point S_p^1 lies within the face while point S_p^2 lies outside the face. If the point lies within the face then the distance to the polyhedral model is the distance d_p.

5) If the point lies outside the face then the shortest distance is to an edge or a vertex of the current face. Determine the edge that returns the largest positive DE_{ij} since the shortest distance must be to that edge or one of the vertices that form the edge.

6) The distance function returns the minimum distance from a surface point to the closest polyhedral model if the surface point is outside and a negative value if the point is inside the model.

The time complexity of the distance computation algorithm developed above is $O(F)$ where F is the number of faces. The distance function returns the distance for a single surface point. The overall proximity cost component is defined as:

$$C_p = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} C_{ij} \tag{3.4}$$

where,

$$C_{ij} = \begin{cases} K_1 & if \quad D_{ij} \le 0 \\ K_2(\delta - D_{ij}) & if \; 0 < D_{ij} \le \delta \\ K_3 f(\delta) & if \quad D_{ij} > \delta \end{cases} \tag{3.5}$$

K_1, K_2 and K_3 are positive constants which are used to control the relative emphasis of C_p with respect to the total configuration cost. The function $f(\delta)$ set to zero in most cases however in cases where the surface must lie within a tolerance region of the polyhedral models an appropriate function may be chosen.

3.1.2. Area Constraint

This constraint is introduced to counteract the expansion effect of the proximity constraint for certain design problems, and is formulated as follows. The surface area of a parametric surface is defined as (Mortensen, 1985):

$$Area\,(S\,(u,v)) \;=\; \iint \left| S^{u}\,(u,v) \times S^{v}\,(u,v) \right| du dv \tag{3.6}$$

and can be calculated approximately via Gaussian quadrature. Thus the area cost component is formulated as,

$$C_{area} \;=\; K_4 Area\,(S\,(u,v)) \tag{3.7}$$

where K_4 is a positive constant similar in effect to the constants associated with the proximity cost.

However preliminary experimentation with this formulation revealed that area evaluations are computationally expensive due to the numerical integration involved [Oliver and Theruvakattil, 1993a]. Consequently, an alternative cost component is introduced which generates a similar effect and is not as expensive computationally. A description of this alternate constraint formulation follows in the next section.

3.1.3. Arc length constraint

In a parametric surface model, if one parameter is held constant while the other parameter is varied through its range, a curve is generated which lies in the surface. A constraint which penalizes the length of these constant parameter curves is formulated. The resulting cost component C_l is thus defined as,

$$C_l = K_5 \{ \sum_{j=0}^{N-1} L_{u_j} + \sum_{i=0}^{M-1} L_{v_i} \} \;=\; K_5 C_{arc} \tag{3.8}$$

where, K_5 is a positive constant which controls the significance of the arc-length cost on a problem and,

$$\begin{aligned} L_{u_j} &= \sum_{i=0}^{M-2} \left| S\,(u_{i+1}, v_j) - S\,(u_i, v_j) \right| \\ L_{v_i} &= \sum_{j=0}^{N-2} \left| S\,(u_i, v_{j+1}) - S\,(u_i, v_j) \right| \end{aligned} \tag{3.9}$$

Note that at each sample point, the area cost requires evaluation of both the parametric derivatives and evaluation of their cross-product, while the arc-length cost is only dependent on surface points.

3.1.4. Parametric flow constraint

In many applications, it is necessary to arrive at an optimum design that maintains the initially rectangular topological structure of the surface. This constraint tends to enforce a relatively uniform relationship between variations in the parameter and the corresponding spatial variations, and is mathematically expressed as,

$$C_{par_u} = K_6 \sum_{i=0}^{M-2} \sum_{j=0}^{N-1} \left| \{ |S(u_{i+1}, v_j) - S(u_i, v_j)| - \frac{L_{u_j}}{(M-1)} \} \right| \tag{3.10}$$

$$C_{par_v} = K_6 \sum_{i=0}^{M-1} \sum_{j=0}^{N-2} \left| \{ |S(u_i, v_{j+1}) - S(u_i, v_j)| - \frac{L_{v_i}}{(N-1)} \} \right| \tag{3.11}$$

where the parametric flow in either parametric direction may be penalized individually or summed to penalize parametric flow constraint dissatisfaction in both parametric direction,

$$C_{par} = C_{par_u} + C_{par_v} \tag{3.12}$$

The parametric flow constraint will have the effect of inhibiting the close proximity of neighboring control points in situations where the control points tend to cross-over. In addition, this constraint will have the effect of driving the design to a spatially uniform design if desired.

3.1.5. Orthogonality constraint

This constraint is imposed in situations where it is desired to have a design such that the constant parameter curves in the u and v directions remain near-orthogonal. This constraint

will also have the effect of preventing the neighboring control points from crossing over. The scalar product of the u and v tangent vectors are used to impose this constraint and a penalty is assessed for non-orthogonality of the tangent vectors, resulting in the orthogonality cost component defined as,

$$C_{ortho} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} C_{or_{ij}} \tag{3.13}$$

where,

$$C_{or_{ij}} = \begin{cases} 0 \text{ if } 0 \le \left| S^u(u_i, v_j) \bullet S^v(u_i, v_j) \right| \le \delta_0 \\ K_8 \text{ if } \left| S^u(u_i, v_j) \bullet S^v(u_i, v_j) \right| > \delta_0 \end{cases} \tag{3.14}$$

The parameter δ_o in Equation (3.14) is a threshold parameter which allows for the specification of the degree of non-orthogonality that may be tolerated without penalty. Experiments indicate that the orthogonality constraint has the desired effect of enforcing a general rectangular topology, but generally not until after the solution has reached the vicinity of the global solution., i.e., its effect is relatively subtle, and noticeable only in the later stages of the solution process.

3.1.6. Curvature constraints

The use of curvature constraints to obtain fair and aesthetically pleasing curves has historical roots in the ship building industry. Given a set of points, how does one smoothly interpolate them? The procedure was to make use of metal weights ("ducks") at the points attached to a wooden beam ("spline") [Farin, 1993]. It so happens that the resulting curve assumes a shape that minimizes the strain energy, and strain energy is related to the curvature. Thus, curvature has played a very important role in the creation of smooth and aesthetically pleasing curves. More recently, analytical curvature plots have been used to smooth surfaces [Farin, 1993].

For a detailed discussion of surface curvatures see Farin [1993]. The curvature of a surface is typically described in terms of the curvature of a curve in the surface. At any point on the surface, it is possible to find a curve passing through the point such that it has a minimum curvature. Likewise it is possible to find a curve that has maximum curvature. The product of the maximum and minimum curvatures yields the Gaussian curvature and the sum yields the mean curvature.

3.1.7. Symmetry constraints

This constraint is motivated by applications in which the symmetry of the final product can be exploited to reduce the size of the problem. This is analogous to a common procedure employed in finite element analysis of symmetric structures. When imposing this constraint, one parametric direction is assigned in the plane of symmetry, and tangent vectors in the other parametric direction are constrained to remain perpendicular to the plane of symmetry. This type of constraint may be applied in both parametric directions simultaneously. The symmetry constraint will be stronger if the curvatures at the plane of symmetry are set to zero. The precise formulation of the symmetry constraint and the associated cost component C_{sym} is application dependent and is a sum of penalties for constraint dissatisfaction in a local region. Associated with this cost component, there is a constant parameter K_{10} which has the same purpose as the constant parameters described earlier.

3.1.8. Manufacturability constraints

Various manufacturability constraints can be formulated in terms of analytic surface properties. One such manufacturability constraint, a formability constraint, has been developed by Nair and Oliver [1993]. However, this constraint has yet to be implemented in a surface synthesis problem.

The constraints developed in this section are by no means exhaustive, indeed any constraint formulated in terms of analytic surface properties can be incorporated into the overall

cost function. Some other possible constraints are briefly discussed in the final chapter of this dissertation.

CHAPTER 4. OPTIMIZATION TECHNIQUES

Given a design environment and specified constraints, a surface that results in the best configuration while satisfying the constraints, is desired. A description of the constraints and their relationship to analytic surface properties are included in the previous chapter. Given an initial surface configuration, how does one proceed to improve it, subject to the applied constraints? The design variables could be manipulated until the best configuration is achieved. Such manual perturbation of the control points is possible but this could turn out to be a rather tedious and time intensive proposition, as the designer would not know when to terminate the process. The need to improve the design, ultimately generating the best design, suggests a systematic procedure for perturbing the design variables, evaluating the corresponding surface configuration and terminating if the best design has been achieved. This is essentially a design optimization problem.

There are several optimization techniques available, so which is the best optimization technique for the problem? The answer depends on the nature of the problem. So, what is the nature of the problem at hand? Is it unimodal or is it multi-modal in nature? Furthermore, should the variables be bounded or unbounded? To answer these questions, five different algorithms have been studied. The five methods chosen for the initial experiments are a quasi-Newton method, Nelder-Mead simplex, conjugate gradient and two different implementations of simulated annealing. The Nelder-Mead simplex algorithm is a direct search method that depends only on function evaluations [Reklaitis et al., 1983]. Quasi-Newton methods mimic the positive features of Newton's method using only first order information. Simulated annealing (SA) is a probabilistic global optimization technique, introduced by Kirkpatrick et al.

[1983] and Cerny [1985] independently. One of the implementations of SA used in this research is that of Corana et al., [1987] with modifications by Goffe et al., [1994]. The second SA algorithm is called adaptive simulated annealing due to Ingber [1989].

Implementations of the three "traditional" multivariate optimization methods, quasi-Newton, Nelder-Mead simplex and conjugate gradient, from the IMSL Math/Library Edition 10.1 were chosen because of its quality and availability. The routines used are DUMPOL (double precision simplex), DUMCGF (double precision conjugate gradients numerical derivatives) and DUMINF (double precision quasi-Newton with numerical derivatives). The versions that make use of numerical derivatives are chosen because closed form derivatives cannot be realized for the objective functions under discussion. A brief description of each method follows.

4.1. Nelder-Mead simplex method

The Nelder-Mead simplex method (a direct search method) is an improved version of the original sequential simplex method of Spendley, Hext and Himsworth [Reklaitis, 1983] which is not to be confused with the simplex method of linear programming. The original simplex method starts with a regular geometric figure called the simplex, consisting of $n + 1$ vertices in an n-dimensional space, which may be defined by the origin and points along each of the n coordinate directions. Once a simplex is available a new simplex may be generated by projecting any chosen vertex by a suitable distance through the centroid of the remaining vertices of the most recent simplex. The function is evaluated at all vertices and the vertex that yields the worst function value is replaced by the projection.

The Nelder-Mead algorithm is different from the original simplex method because it allows for expansions and contractions in the course of the reflection step. Given an initial simplex, the objective function $f(\boldsymbol{x})$ is evaluated at each of the vertices $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

Denote the vertices where the function assumes its maximum and minimum values as x_h and x_l respectively. As described earlier the simplex method replaces the vertex x_h by a point where the function has a lower value. The new point x_r is arrived at by the following formula:

$$x_r = x_c + \alpha (x_c - x_h) \tag{4.1}$$

where

$$x_c = \frac{1}{n} \sum_{i \neq j} x_i \tag{4.2}$$

and α $(\alpha > 0)$ is the reflection coefficient. If it happens that the new vertex x_r is such that $f(x_r) \leq f(x_i)$ for $i = 1, 2, \ldots, n+1$, an expansion is attempted based on the following formula:

$$x_e = x_c + \beta (x_c - x_r) \tag{4.3}$$

where β $(\beta > 1)$ is the expansion coefficient. If the point x_r is a worse point, then a contracted point x_c is computed. If the contraction step is unsuccessful, the size of the simplex is reduced by moving the vertices half-way toward the current best point. This process is continued until one of the following stopping criteria is satisfied:

$$f_{best} - f_{worst} \leq \varepsilon_f (1 + |f_{best}|) \tag{4.4}$$

or

$$\sum_{i=1}^{n+1} \left(f(x_i) - \frac{\sum_{j=1}^{n+1} f(x_j)}{n+1} \right)^2 \leq \varepsilon_f \tag{4.5}$$

where ε_f is a given tolerance.

4.2. Conjugate gradient method

The conjugate gradient method is a first order technique for unconstrained minimization. The main advantage of the conjugate gradient method is that it provides a fast rate of convergence without the need to store any matrices, thus making it useful for solving problems with many variables. The IMSL routine, DUMCGG, uses the algorithm described by Powell [1975]. A brief description of the algorithm follows.

1. Given x_0, define s_0 to be the steepest descent direction,

$$s_0 = -\nabla f(x_0) = g_0 \tag{4.6}$$

let the counters k and t be set to 0. and begin iterations by incrementing k.

2. For $k \geq 1$, the direction s_k is defined by:

$$s_k = -g_k + \beta_k s_{k-1} + \gamma_k s_t \tag{4.7}$$

and

$$g_k = -\nabla f(x_k) \tag{4.8}$$

where

$$\beta_k = \frac{g_k^T [g_k - g_{k-1}]}{s_{k-1}^T [g_k - g_{k-1}]} \tag{4.9}$$

and

$$\gamma_k = \frac{g_t^T [g_{t+1} - g_t]}{s_t^T [g_{t+1} - g_t]} \quad \text{if } k > t+1$$
$$\gamma_k = 0 \qquad \text{if } k = t+1 \tag{4.10}$$

3. For $k \geq 1$, test the inequality

$$\left| g_{k-1}^T g_k \right| \geq 0.2 \| g_k \|^2 \tag{4.11}$$

A restart is initiated when the inequality holds because that means sufficient orthogonality between g_{k-1} and g_k has been lost. Accordingly, t is reset $t = k-1$ to imply a restart.

4. For $k > t+1$ the direction s_k is also checked, to guarantee a sufficiently large gradient, by testing the inequalities

$$-1.2\|g_k\|^2 \le s_k^T g_k \le -0.8\|g_k\|^2 \tag{4.12}$$

If these inequalities are not satisfied, the algorithm is restarted by setting $t = k-1$.

5. The algorithm is also restarted if $k - t \ge n$, by setting $t = k-1$.

6. The process is terminated if $\|g_{k-1}\|$ or $|f(x_{k+1}) - f(x_k)|$ is sufficiently small. If not, k is incremented by one and the process is repeated by going to step 2.

4.3. Quasi-Newton method

Quasi-Newton methods belong to the family of gradient methods. In the Newton method the descent direction $s(x_k)$ at the current estimate of the optimum is: $s(x_k) = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$. The main disadvantages of the Newton method are that for every iteration, Newton's method requires calculation of the Hessian and solution of a system of linear equations to compute the inverse of the Hessian, both of which are computationally intensive. In addition, the method fails to converge from a "poor" initial guess. These negative aspects of the Newton method led to the development of methods known as quasi-Newton which use the gradient information to construct approximations of the Hessian matrix and the inverse Hessian. Another improvement of the quasi-Newton method over the standard Newton method is the use of a line search at each iteration to determine the step size as opposed to maintaining a unit step size.

A typical quasi-Newton algorithm with an inverse Hessian update may be stated as:

$$x_{k+1} = x_k - \alpha_{k+1} s_k \tag{4.13}$$

where

$$s_k = -G_k \nabla f(x_k) \tag{4.14}$$

Where G_i is the inverse Hessian at iteration i and $\nabla f(x_k)$ is the gradient. The IMSL routine, DUMINF, uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) scheme for updating the inverse Hessian matrix. Define $\Delta g = \nabla f(x_{i+1}) - \nabla f(x_i)$ and $\Delta x_i = x_{i+1} - x_i$. The BFGS update formula is:

$$G_{i+1} = G_i - \frac{\Delta x (G_i \Delta g)^T}{\Delta x^T \Delta g} - \frac{G_i \Delta g (\Delta x^T)}{\Delta x^T \Delta g} + \left(1 + \frac{\Delta g^T G_i \Delta g}{\Delta x^T \Delta g}\right) \frac{\Delta x \Delta x^T}{\Delta x^T \Delta g} \tag{4.15}$$

After every iteration the convergence criteria $\| g(x_i) \| = \varepsilon_g$ is checked, where ε_g is the gradient tolerance. Since explicit gradient information is not available for the problems being addressed in this dissertation, a finite difference approximation of the gradient is calculated.

4.4. Simulated annealing

In this method, the cost values of the sequence of solutions is generally decreasing but not strictly, i.e. function values are allowed to occasionally take on increasing costs. The name is derived from the annealing process of metals. In the physical process of annealing, a material is heated and allowed to cool slowly, at a rate such that it reaches thermal equilibrium at each temperature. Consequently, its atoms will reach a state of minimum energy, the ground state, despite any local minima. If the material is cooled rapidly the atoms are likely to freeze at a higher-than-minimum energy level. Simulated annealing has been applied to a variety of combinatorial optimization problems including, VLSI circuit design [Devadas and Newton, 1987], structural truss design [Elperin, 1988], mechanism synthesis [Jain and Agogino, 1988],

robotic path planning [Sandgren and Venkataraman, 1989] and simulation optimization to determine optimal parameter levels at which to operate a system [Haddock and Mittenthal, 1992].

In SA, the objective (cost) function to be minimized is analogous to the total energy of the system. From an algorithmic point of view, SA is essentially an iterative improvement strategy augmented by a criterion for occasionally accepting higher cost configurations [Rutenbar, 1989]. Given an objective (cost) function $f(x)$ and an initial state vector x_0, iterative improvement seeks to improve the current state by randomly perturbation of x_0. If the new state x_i yields a lower cost, then it replaces the current state and the perturbation process continues from x_i. If the perturbed state produces a higher cost than the original state it is rejected, and the perturbation continues from the original state. This procedure is continued until no further improvement in the cost can be obtained. The drawback of iterative improvement is the possibility of converging to a local minimum. One could restart the process using a number of different initial configurations and take the best result, yet there is no guarantee that the optimal solution will be found. In addition, such an approach can be extremely inefficient computationally.

If a higher cost state is generated in the SA algorithm, the state is accepted with a probability based on the current temperature. Since the probability of accepting a higher cost state decreases with temperature, the SA algorithm mimics the physical process of annealing. It has been described as a "hill-jumping" technique because it can tolerate temporary degradation of the objective function to "jump" out of the vicinity of a local minimum. The behavior of the SA algorithm has been characterized as, first following the gross behavior of the objective function to find an area in its domain where the global minimum should be present, irrespective of local minima found on the way. It progressively develops finer details, ultimately finding a good, near-optimal local minimum if not the global minimum itself [Corana, et al., 1987;

Laarhoven and Aarts, 1987]. Theoretical studies related to simulated annealing are reported by Laarhoven and Aarts, [1987], Romeo and Sangiovanni-Vincentelli [1991] and Dekkers and Aarts [1991].

In essence, simulated annealing presents an optimization technique that can:

- accommodate objective functions with arbitrary degrees of nonlinearities, discontinuities, and stochasticity;
- process arbitrary constraints and boundary conditions imposed on the objective functions;
- be easily implemented relative to other nonlinear optimization techniques;
- statistically guarantee finding a globally optimal solution.

The main criticism facing standard SA is that it is time-consuming to find the optimal solution. Another negative feature is the difficulty of fine tuning the SA parameters for specific problems relative to traditional optimization techniques. Often the selection of the appropriate starting temperature and cooling rate are settled by trial and error. In any practical design tool, it is necessary to provide capabilities that are easily understood by designers. Consequently, a cooling schedule that is not heavily problem dependent is desired. Even if it is problem dependent, the designer must be in a position to change some of the cooling schedule related parameters without much experience with simulated annealing. In essence, SA algorithms are sought so as to provide the designer of sculptured surface models with a robust optimization procedure. Romeo and Sangiovanni-Vincentelli [1991] suggest using an adaptive algorithm because such algorithms make use of runtime information to modify some of the SA parameters. Thus, the user does not have to fine tune the SA parameters by time consuming experimentation. Romeo and Sangiovanni-Vincentelli [1991], also mention that the performance of dynamic/adaptive SA schemes are superior to static schemes in all respects, except probably simplicity. Oliver and Theruvakattil [1993b] use a static scheme for the synthesis of sculptured surface models. Both the SA algorithms used in this research are adaptive

and require very little fine-tuning of parameters.

4.4.1. Simulated annealing for continuous variable problems

Vanderbilt and Louie [1984] demonstrated that simulated annealing may be equally well applied to continuous parameter problems by applying it to a set of "standard" optimization problems and to a functional fitting problem. Bohachevsky et al. [1986] used simulated annealing for function optimization. Corana et al., [1987] introduced a SA algorithm with the capability of making adaptive moves along the coordinate directions and some aspects of this algorithm were later improved by Goffe et al., [1994]. Ingber, [1989] introduced a version of SA called very fast simulated re-annealing (VFSR) which can be used for continuous or discrete variable problems. Several other versions of SA have been suggested and are in use within the research community. However, for this research the latter two adaptive algorithms are employed.

4.4.2. Algorithm 1 (SA1)

The first version of SA used in this research is due to Corana et al., [1987], with modifications by Goffe et al., [1994] and is chosen due to ease of use and robustness. The source code for this implementation of SA is made available by Goffe [1994].

In this algorithm the user has control over the starting temperature T_0, the initial guess of the parameters $\boldsymbol{x}$ and the step length of the parameters $\boldsymbol{s}$. The vectors $\boldsymbol{x}$ and $\boldsymbol{s}$ are of length n, the number of parameters. A brief description of the algorithm follows; for details see Corana et al., [1987] and Goffe et al. [1994].

The user specified starting point $\boldsymbol{x}$ is used to determine the initial objective function value, $f(\boldsymbol{x})$. Then a new point $\boldsymbol{x}'$ is chosen by varying the i-th element of $\boldsymbol{x}$,

$$x'_i = x_i + r \cdot s_i \tag{4.16}$$

where r is a uniformly distributed random number in [-1, 1] and s_i is the ith element of $\boldsymbol{s}$.

The resulting objective function value, $f(x')$, is computed and if it is less than $f(x)$, x' is accepted and x' is set to x (a downhill move has been accepted). The best current minimum objective function value and associated point x are also recorded.

If the new objective function value $f(x')$ is greater than or equal to $f(x)$ then the Metropolis criteria [Metropolis et al., 1953] is used to decide acceptance. Let $\Delta f = f(x') - f(x)$ then the new state is accepted with probability $P = exp(\frac{-\Delta f}{T})$. The acceptance criteria is implemented by comparing P to P' a uniformly distributed random number from [0,1] such that if P is greater than P', the new point is accepted and x is updated with x' (an uphill move has been accepted). Otherwise x' is rejected and a new x' is generated from x. The probability of accepting an uphill move decreases with lower temperatures and with larger Δf.

There is an inner and outer loop associated with SA. For this algorithm the inner loop is controlled by N_s. After N_s steps through all elements of x the step length vector s is modified such that approximately 50% of all moves are accepted, thus allowing the function to be sampled widely. The argument for maintaining 50% acceptance is that if it is less then the steps being taken are too big, thus wasting computational effort. On the other hand, if the acceptance rate is more than 50% then the steps are too small and therefore the function is not being sampled widely enough. For details on how s is modified to achieve 50% acceptance see Corana et al., [1987]. As the temperature decreases, the acceptance probability reduces and this in turn reduces the step length.

The outer loop controls the temperature. After N_T cycles through the inner loop, the temperature, T is reduced as follows:

$$T' = r_T \cdot T \tag{4.17}$$

where r_T is a number between 0 and 1. As the temperature decreases, uphill moves become less likely, thus increasing the number of rejections which then leads to a reduction of the step lengths. At the new temperature iterations, begin from the current optimum thus focusing on the most promising area. If the objective function value is within a user specified tolerance ε over N_ε consecutive temperature cycles, it terminates.

Goffe et al., [1994] have made some improvements to the algorithm outlined in this section. The first modification allows the user to rerun the algorithm with a different starting value and a different seed for the random number generator. This has the effect of generating a completely different solution sequence. Thus the user will have more confidence in the final result if the two solution sequences yield the same solution. Another modification allows the user to determine the appropriate starting temperature by setting $r_T > 1$ and checking the components of the step length vector s. The temperature that results in s (step length) components which can sample the function widely, is used as the starting temperature. A third modification allows the user to restrict the optimization to a subset of the parameter space. This allows for better understanding of the function.

4.4.3. Algorithm 2. Adaptive simulated annealing (ASA)

Adaptive simulated annealing (ASA) [Ingber, 1993,1994] was also called very fast simulated re-annealing (VFSR) [Ingber, 1989]. Ingber argues, that in a given optimization problem, the different parameters have different finite ranges. These finite ranges on the parameters are fixed by physical considerations and different annealing-time dependent sensitivities which are measured by the curvature of the cost function at a local minimum. Most of the other versions have distributions that sample infinite ranges, and there is no provision for considering differences in each parameter dimension. To this effect, Ingber [1989] introduces the idea of different annealing schedules for the parameters depending on the sensitivities,

measured by the curvature of the objective function at the current best solution. However, it must be noted that SA1 is also adaptive and allows for physical bounds on the variables. Some of the important aspects of this algorithm are outlined in the following paragraph. However, the reader is directed to Ingber [1989,1993,1994] for a detailed description.

Let $\boldsymbol{x}_k$ be the vector of parameters generated at annealing-time k. Then, the element x_k^i in dimension i with $x_k^i \in [A_i, B_i]$ is involved in the computation of x_{k+1}^i as follows:

$$x_{k+1}^i = x_k^i + y^i (B_i - A_i) \tag{4.18}$$

where y^i is a random number in $[-1, 1]$ based on the following distribution;

$$y^i = \text{sign}\,(u^i - \frac{1}{2})\, T_i \left[(1 + \frac{1}{T_i})^{|2u^i - 1|} - 1 \right] \tag{4.19}$$

where u^i is from the uniform distribution $u^i \in [0, 1]$. Figure 4.1 shows the distribution of the random number y^i as function of the parameter temperature and the random number u^i. Figure 4.1 shows that as the parameter temperature decreases the most likely choice of random number y^i is close to 0. This explains how the step size (see Equation (4.18)) is controlled as the parameter temperature decreases.

The annealing schedule for the parameter temperature and the cost acceptance temperature are both exponential, given by:

$$T_i(k) = T_{01} exp\left(-c_i k^{\frac{1}{D}}\right) \tag{4.20}$$

For the parameter temperature the number of generated points is used to determine k while the number of accepted points is used for annealing the cost temperature. c_i is controlled by two variables m_i and n_i (user inputs) that can be used to help tune ASA for specific

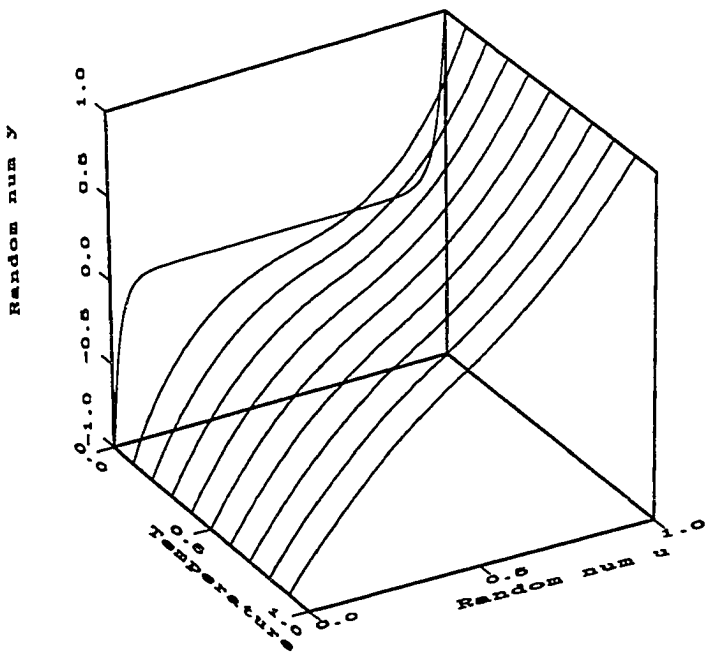


Figure 4.1 Random number y as a function of temperature and random number u

problems. The annealing (cooling) schedule for ASA is exponential, which is faster than that of SA1.

The ASA algorithm perturbs all the parameters before evaluating the objective function whereas, the SA1 algorithm evaluates the objective function after perturbing one parameter at a time. During the perturbation process, ASA keeps generating random numbers, y^i, until the perturbed point is within the specified bounds. SA1 generates a new parameter value from the current value using the step length, the size of which is related to the cost temperature. The ASA source code was made available by Dr. Lester Ingber [Ingber, 1994] and adapted for the task of improved surface design. ASA has several control parameters at the disposal of the user, however, the most important of these are:

- T_{0i} initial parameter temperatures

- *TRS* temperature ratio scale
- *TAS* temperature annealing scale

The method of ASA, has been applied to a variety of problems. Some of the early applications of ASA include solutions to problems in combat analysis, finance, and neuroscience most of which is attributable to Ingber [1993]. There have been some recent applications of ASA by Wu and Levine [1993] for 3-D object representation using parametric geons. They use ASA to obtain a unique model while compensating for noise and minor changes in object shape. Blais and Levine [1993] formulate the problem of range image registration as an optimization problem and solve it with ASA. Range image registration is the process of finding the translation and rotation parameters which properly align overlapping views of the object, thus enabling reconstruction of an integral surface representation of an object from partial surfaces.

4.4.4. Genetic algorithms (GA)

GA's are a popular class of algorithms, competitive with simulated annealing in the area of global optimization. GA's are search strategies based on the mechanics of natural selection, first introduced by Holland [1973]. Michalewicz and Janikow [1991], address global optimization of continuous variable functions using GA's and some of the associated problems such as; premature global convergence, rapid local convergence, and the handling of constraints. Goffe et al., [1994] reported difficulty optimizing simple continuous variable problems using GA's. Michalewicz and Janikow [1991], and Goffe et al., [1994] indicate that GA's are in need of further development for continuous variable functions.

4.5. Summary

In this chapter, a brief description of all the optimization methods under consideration for solving the surface synthesis problem, is provided. The advantages and disadvantages of

some of the techniques have been outlined. In the following chapter two examples are used to discriminate between the different optimization techniques, ultimately settling for one of the methods.

CHAPTER 5. INITIAL EXAMPLES

The primary objective of the examples presented in this chapter are to test the constraints developed in Chapter 3, and to determine the most suitable optimization method among those discussed in Chapter 4. The first example is a simple surface in the vicinity of a single polyhedral model. The second example is similar in nature but has three polyhedral models in the vicinity of the surface. All the computations are carried out on a Silicon Graphics, Onyx, workstation.

5.1. Example 1

5.1.1. Problem description

The problem configuration consists of a single surface in the vicinity of a single polyhedral model in an initial orientation as shown in Figure 5.1. The configuration shown in Figure 5.1 is only one of many different starting configurations attempted. The surface model used is a non-rational bi-cubic B-spline surface with four control points in each parameter direction (i.e., a Bezier surface). The control points on the edges of the surface are constrained (fixed), while the positions of the four interior control points serve as the design variables. Each control point gives rise to three design variables since the control points can be perturbed in three independent coordinate directions. There are a total of 12 design variables for the system. The surface is evaluated at 25 points in the u and v parameter directions, i.e., $M = N = 25$. Four constraints are applied to the design environment, the proximity constraint, arc length constraint, orthogonality constraint and parametric flow constraint (see Chapter 3). The arc length constraint is applied in place of the area constraint because it has

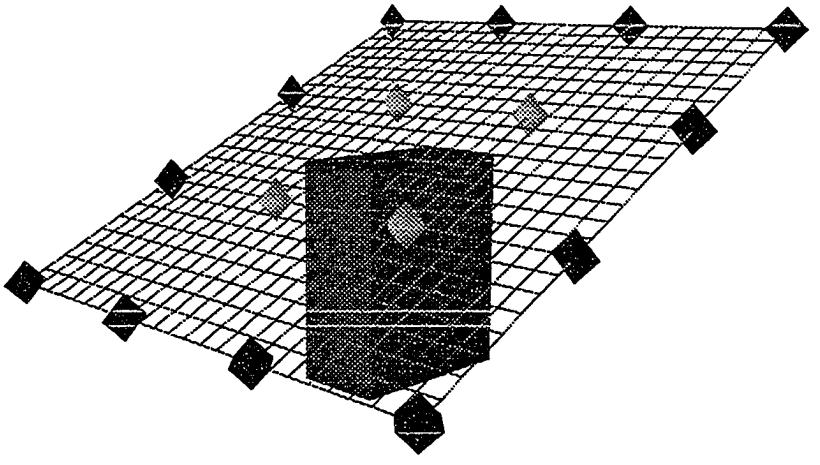


Figure 5.1 Initial configuration of the surface in the vicinity of a single polyhedral

the same effect and is computationally less expensive [Oliver and Theruvakattil, 1993]. The values for cost penalties used are: $K_1 = 150.0$, $K_2 = 300.0$, $K_3 = 0.0$, $K_5 = 5.0$, $K_6 = 2.0$ and $K_7 = 10.0$. The functional form of the proximity cost for a single surface point is as shown in Figure 5.2., indicating that the first derivative of the overall objective function may not be continuous.

The surface synthesis problem is started from different configurations for this example because the standard optimization techniques are sensitive to starting configurations. Various starting configurations are generated by randomly perturbing the control points of an initially flat surface as shown in Figure 5.1. Ideally, many starting configurations would have to be attempted however, for this study only nine different starting configurations were attempted. With the first version of simulated annealing (SA1) all the 9 starting configurations are attempted. For ASA only one starting configuration is attempted but several executions of the algorithm are made with different ASA parameters.

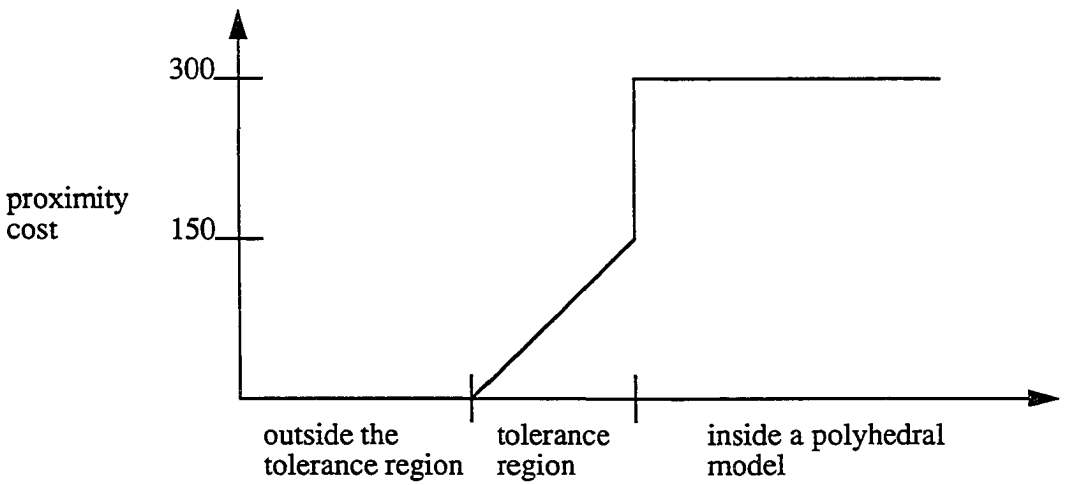


Figure 5.2 Proximity cost as a function of distance from the polyhedral model

5.2. Results for example 1

5.2.1. Nelder-Mead simplex

For all the runs the tolerance is set to $\varepsilon_f = 1.000\times10^{-6}$ and the size of the initial simplex is set to 5 units. The results for the 9 different initial configurations are shown in Table 5.1. Various sizes of the initial simplex, ranging from 1-10 units did not improve the final

Table 5.1 Results from the 9 different initial configurations (Nelder-Mead)

Initial config.	final cost	# of function evals.	time (secs)
1. Flat surface	64.358	3,229	755
2	75.738	1,826	385
3	65.009	1,697	362
4	69.823	1,348	285
5	85.338	2,007	1,563
6	68.239	2,166	477
7	66.407	2,257	492
8	69.275	1,589	335

Table 5.1 Results from the 9 different initial configurations (Nelder-Mead)

Initial config.	final cost	# of function evals.	time (secs)
9	68.016	1,924	410

objective function value. Only the run with an initially flat surface results in a final symmetric location of the control points as shown in Figure 5.3. All the other trials result in non-symmetric control point location. For this particular problem, it is expected that the final orientation of the control points is symmetric.

On closer inspection of the "time" column of Table 5.1, it is obvious that the time

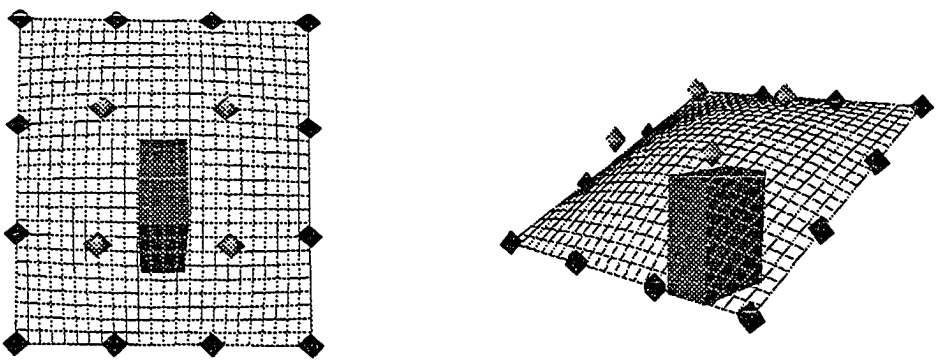


Figure 5.3 Final configuration (Nelder-Mead)

taken is not consistent with the number of function evaluations. This is due to varying system load on the computer during execution of the runs.

5.2.2. Conjugate gradient method

For the conjugate gradient method the convergence criteria is set to $\varepsilon_f = 1.000\times10^{-6}$. For all the nine runs the conjugate gradient algorithm terminated with the following report, "The line search of an integration was abandoned. An error in the gradient may be the cause." Goffe et al., [1994] reported similar problems while using the conjugate gradient algorithm. A

possible cause of this error is very large gradient components resulting from certain surface configurations or large gradients as a result of the finite difference approximations. The final objective function values range from 65.529 to 73.542. The number of function evaluations at termination range from 783-1794. As is the case with Nelder-Mead, only the initially flat surface results in symmetric control point locations at the time of termination.

5.2.3. Quasi-Newton method

The quasi-Newton method returned identical results regardless of the starting configuration. The convergence tolerance for this method is set to $\varepsilon_g = 6.056\times10^{-6}$. Table 5.2 shows some of the computational statistics for the 9 runs. All the runs result in identical, symmetric

Table 5.2 Quasi-Newton results

final cost	average time in seconds	avg. # of iterations	avg. # of functions evals.	avg. # of gradient evals.
63.799	229	55	172	69

placement of the control points. The final objective function values are all identical. While observing the solution process it was noticed that the first big change in the surface configuration results in a surface that "balloons up", avoiding the polyhedral model. This could be due to the first line search yielding a rather large step size that takes the surface away from the polyhedral model. So, from that configuration of the surface the quasi-Newton algorithm leads it to the final configuration along a smooth and continuous hill-side.

5.2.4. Simulated annealing (SA1)

The tolerance for SA1 is set to $\varepsilon_f = 1.000\times10^{-6}$. As a first test, conservative values for the parameters are used as suggested by Corana et al., [1987]. Some of the pertinent parameters are: $T_0 = 50.0$, $N_T = 100$, $r_T = 0.85$ and all elements of s are set equal to 5

units. Given that the control points are bounded within a cube of dimension 20 units it is assumed that the initial value of s is large enough to visit the entire solution space. This run required 2, 904, 001 function evaluations and resulted in a final objective function value of 63.799 with symmetric control points just as in the case of the quasi-Newton runs. However, the number of functions evaluations for this rather simple problem is prohibitively high, so it was decided to try a run with less conservative parameters as suggested by Goffe et al [1994]. Consequently, N_T, and r_T are set to 10 and 0.5 respectively. This run resulted in an objective function value of 63.854 using 76747 function evaluations. Another run with a different seed value resulted in an objective function value of 63.807 requiring 79160 function evaluations. These two runs give sufficient confidence in the less conservative parameters. The runs with

Table 5.3 SA1 results

run #	cost function value	# of function evaluations
1 (initially flat surface)	63.807	79,160
2	63.860	76,614
3	63.987	78,363
4	63.831	74,816
5	63.813	76,763
6	63.929	72,152
7	63.811	74,444
8	63.821	75,620
9	63.808	76,912

different starting configurations are all made using the new set of less conservative parameters. The results, including one of the runs described earlier are shown in Table 5.3. Figure 5.4 shows the trajectories of the objective function from the initial to final configuration for the first five runs shown in Table 5.3. As is obvious from Figure 5.4, different initial configura-

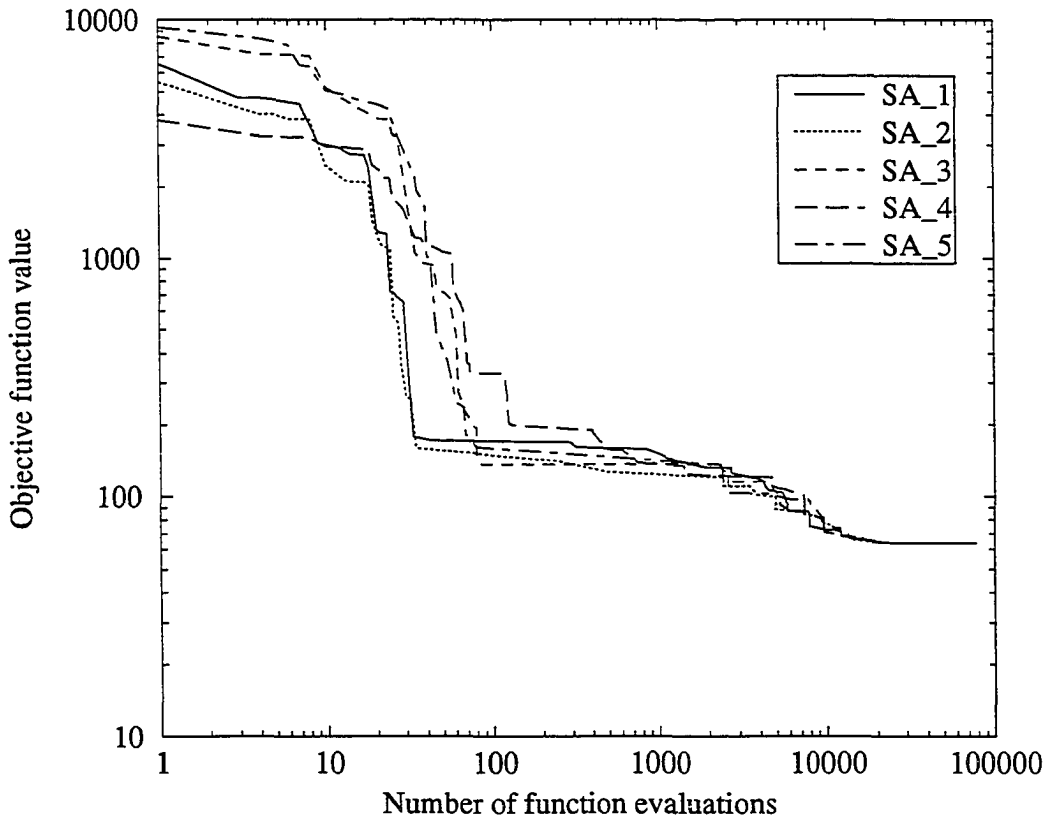


Figure 5.4 Objective function trajectory for 5 different SA1 runs

tions and different seeds for the random number generator result in completely different trajectories.

5.2.5. Adaptive simulated annealing (ASA)

In general ASA requires the specification of bounds on the control points. For the first run of the ASA algorithm the bounds are set the same as for SA1 (rather large bounds). The main ASA parameters are set as follows based on numerous experiments: TRS = 1e-05, TAS = 100, and T = 1.0. This run results in an objective function value of 66.147 requiring 216,680

function evaluations. The final location of the control points are not symmetric. The bounds on the control points are tightened, resulting in an objective function value of 64.454 in 209,104 function evaluations. Even though the final placement of the control points is symmetric, the actual location is noticeably different from that of the quasi-Newton and SA1 runs. To be consistent when making comparisons to SA1, it was decided to run ASA with the same bounds as SA1. All the runs are started from the initially flat surface with different ASA parameters. Five runs were made with different ASA parameters resulting in final objective function values ranging from 66.120 to 74.195. None of the five runs result in completely symmetric placement of the control points. The average number of function evaluations is 142,547. It is also observed that the time required per function evaluation for the ASA algorithm is almost twice that of the SA1 algorithm due to the method of perturbing all the control points and the need to keep track of derivative information for reannealing.

5.2.6. Discussion of results

On inspection of the final solutions due to the different methods, it is obvious that the quasi-Newton method yields the best solution in the shortest period of time. The SA1 algorithm also gives good results. Both, the quasi-Newton and SA1 algorithms reach the best possible solution regardless of the starting configuration. The SA1 algorithm requires significantly more time to reach a final solution especially when using the conservative parameters. Even with the use of the less conservative parameters, SA1 requires many more function evaluations. However, the performance of SA1 is compares well with performances reported by Corana et al., [1987] and Goffe et al, [1994].

The Nelder-Mead algorithm always converged to a different solution depending on the starting configuration. Even the best solution offered by the Nelder-Mead method is not as good as any of the quasi-Newton or SA1 solutions. All the runs involving the conjugate gradient method terminated early. This is probably due to very large components in the gradient

vector which is possible when the surface takes on certain configurations or due to finite difference approximation errors. The ASA algorithm always converged to a greater cost than quasi-Newton or SA1. A number of different ASA parameter combinations were attempted without success. The ASA algorithm performs poorly because the adaptive feature is not as directly related to the objective function as that of the SA1 algorithm. While the step length in SA1 is controlled by the cost temperature (the control parameter usually associated with simulated annealing), the step length in ASA is controlled by the parameter temperature which is not directly related to the cost temperature. Another possible cause for the poor performance of ASA is the perturbation of all the control points before evaluation.

The results of the Nelder-Mead runs suggest a multi-modal objective function. Generally, for a multi-modal problem the quasi-Newton method is sensitive to starting values. However, as explained earlier, the first line search during execution of the quasi-Newton algorithm appears to take the surface to a "hill-side" from which the path to the best configuration is without any local optimums. Therefore, it was decided to attempt a more challenging problem with a few easily identifiable locally good solutions. However, the problems encountered with the conjugate gradient method and ASA for this simple problem takes them out of consideration for further experiments. Even though the Nelder-Mead technique failed to converge to the same solution as quasi-Newton or SA1, it will be used to synthesize the next problem along with the quasi-Newton and SA1 algorithms.

5.3. Example 2

5.3.1. Problem description

The second design environment consists of three polyhedral models and an initial surface as shown in Figure 5.5. The surface is a non-rational B-spline, with seven control points in both parameter directions. All the control points along the edge of the surface are fixed during the synthesis while the interior control points are free to change position. Thus, this prob-

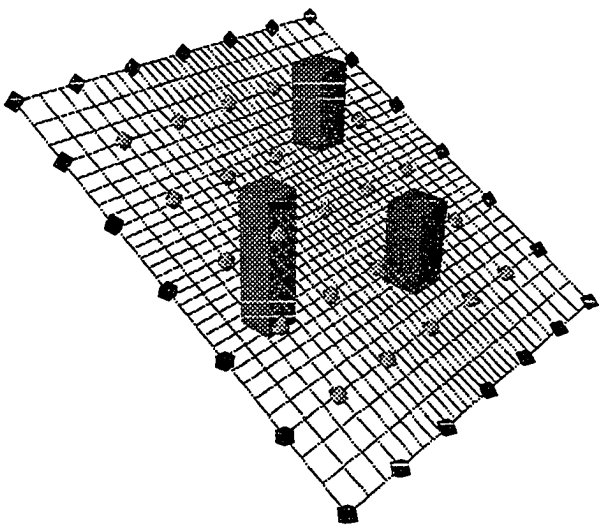


Figure 5.5 Initial configuration of surface with three polyhedral models

lem has a total of 75 degrees of freedom. The surface is evaluated at 30 points in the u and v parameter directions, i.e., $M = 30$ and $N = 30$. Four constraints are applied to the design environment: the proximity constraint, arc length constraint, orthogonality constraint and parametric flow constraints (Chapter 3). The values used for cost penalties are $K_1 = 50.0$, $K_2 = 500.0$, $K_3 = 0.0$, $K_5 = 1.0$, $K_6 = 0.5$ and $K_7 = 5.0$. Due to heavy computational requirements this problem is attempted with only two different starting configurations.

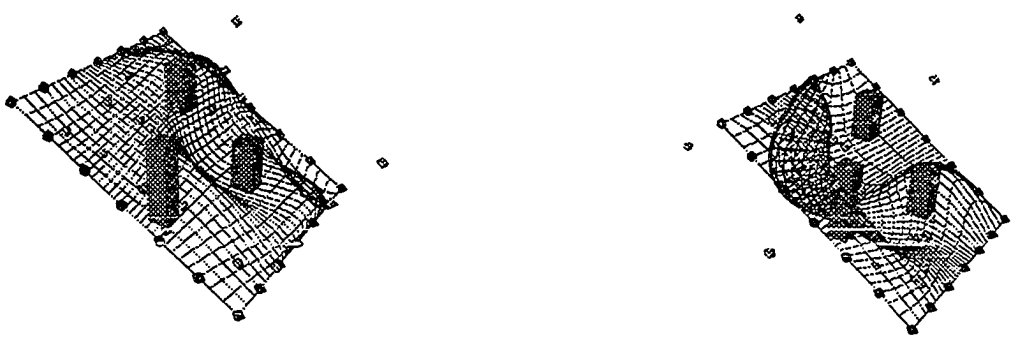
5.3.2. Results - Nelder-Mead simplex

The convergence parameter is set the same as for the previous set of runs. Two runs with different initial surface configurations are attempted. The first run is started from a flat surface as shown in Figure 5.5. For the second run the initial surface is generated by randomly perturbing the control points of the flat surface. The first run converged to an objective function value of 7, 149.54 with 13, 837 function evaluations. The final surface interferes with

two of the three polyhedral models and therefore is not a good solution. For the second run an objective function value of 3, 667.28 is arrived at in 14, 042 function evaluations. In this case the final solution interferes with one polyhedral model.

5.3.3. Results - quasi-Newton method

The convergence parameter for the quasi-Newton method is the same as for the previous example. The two initial configurations are the same as for Nelder-Mead. The first run converged to an objective function value of 1269.102 using 468 function evaluations. The second starting configuration converged to an objective function value of 2035.814 using 157 function evaluations. The final solutions are shown in Figure 5.6((a) is the solution of the first run and (b) that of the second run). On inspection of Figure 5.6 it is apparent that the final sur-



(a) (b)

Figure 5.6 Final configuration of surface with three polyhedral models (quasi-Newton)

face configurations are not desired because of interference with the polyhedral models. Figure 5.6b shows a twisted and warped surface. This happens because there are no bounds on the control points when using the quasi-Newton algorithm thus allowing the control points to cross-over.

5.3.4. Results - SA1 algorithm

The initial configurations are the same as those used for the Nelder-Mead runs. The convergence criteria is the same as the previous SA1 runs. The less conservative parameters used with the previous example are used for this example. However, the seeds for the random number generator are different for the two runs. The first run converged to a cost value of 199.967 in 522, 646 iterations while the second run converged to 200.870 in 554, 331 iterations. An additional run with reduced initial temperature, $T_0 = 20.0$, and increased inner loop iterations, $N_T = 100$. This is essentially a more conservative run to check the results of the first two. The third run converged to 199.935 in 1, 014, 688 iterations. The results of the three different runs give added confidence with regard to the less conservative parameters. The third run required almost double the number of iterations to converge to a solution yet, the improvement in comparison to the other two runs is less than 0.5%. The trajectories of the three runs are shown in Figure 5.7. The final configuration of the surface for the three runs is shown in Figure 5.8.

5.3.5. Discussion of results

Both, Nelder-Mead and quasi-Newton converge to undesirable surface configurations while all the SA1 runs converge to the same final configuration. The results of the Nelder-Mead and quasi-Newton runs further reinforce the fact that the surface synthesis problems are multi-modal in nature. The SA1 algorithm appears to successfully find a "good" solution regardless of "less good" solutions offered by the design space. Since two runs of the SA1 algorithm with different initial configurations and random number seeds converge to the same solution it gives confidence in the result, because the trajectories taken are completely different (see Figure 5.4 and Figure 5.7).

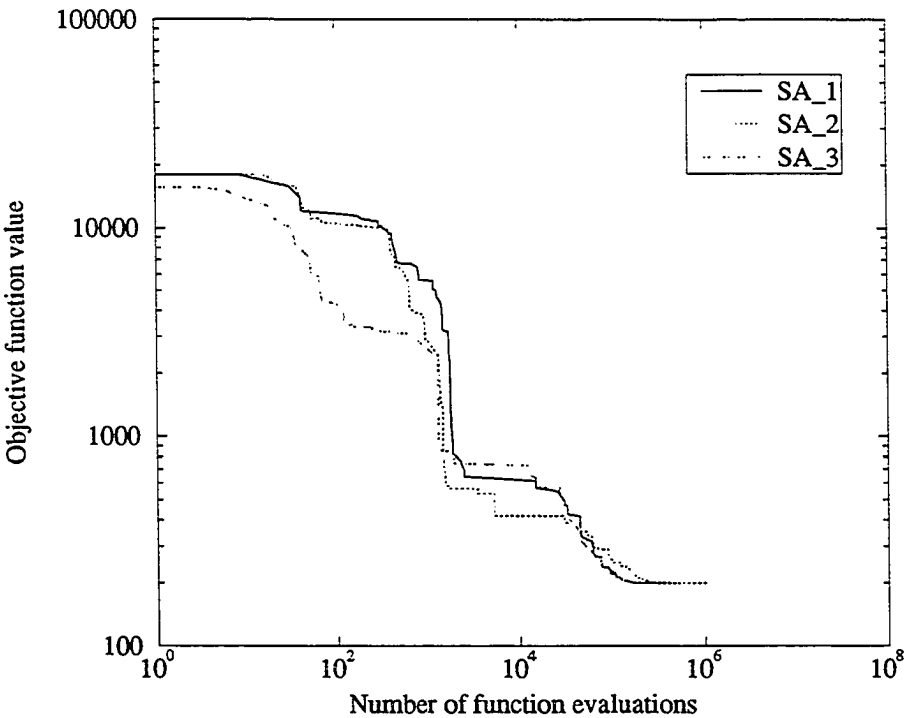


Figure 5.7 SA1, objective function trajectory for example 2

5.4. Summary and conclusions

In this chapter several optimization techniques are attempted on two different example problems. After using the different optimization methods to solve the first problem, it is concluded that the conjugate gradient method and ASA should not be used for the second example due to convergence problems. Even though the Nelder-Mead method does not perform well it is used to find a solution to the second problem. The quasi-Newton method and SA1 always converges to the best solution regardless of starting configuration. The results of the Nelder-Mead method suggest a multimodal problem. However, the quasi-Newton results are to the contrary. It is was then decided to use Nelder-Mead, quasi-Newton and SA1 to solve a more difficult problem with easily identifiable locally good solutions.

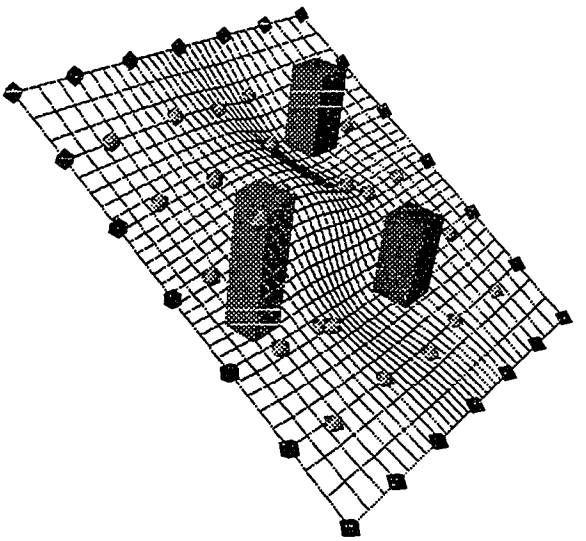


Figure 5.8 Final configuration of surface with three polyhedral models

The results of the Nelder-Mead and quasi-Newton methods are not acceptable surface configurations thus, reinforcing the notion that the surface synthesis problems are multi-modal in nature. However, SA1 converges to the same final solution in all three experiments. The experiments with SA1 have also revealed that relaxation of certain parameters reduces the number of function evaluations significantly, without affecting the final solution.

The results of experiments in this chapter demonstrate the need for an optimization technique that will yield a near optimal final solution. Of all the optimization methods used only SA1 exhibits the capability for avoiding locally good solutions. Consequently, SA1 is the choice for any further surface synthesis problems. In the following chapter, results of some additional examples with SA1 are reported.

CHAPTER 6. ADDITIONAL EXAMPLES

The examples of the previous chapter were primarily used to test the constraint formulation of Chapter 3 and to determine which of the methods presented in Chapter 4 is best suited for sculptured surface synthesis. The results of the examples dictate the use of simulated annealing (SA1 algorithm), because of the multimodal nature of the surface synthesis problems. In this chapter a few additional examples are presented. In each one of these examples a new constraint or feature is introduced. All the problems in this chapter are solved via SA1.

6.1. Surface with free knots

This design environment and the constraints are identical to the second problem presented in the previous chapter. However, in this case the interior knots of the B-spline surface are considered design variables. The surface has three interior knots in each parametric direction therefore, the problem now has 81 design variables as opposed to 75. The initial location of the knots on the surface are as shown in Figure 6.1. On inspection of the surface in Figure 6.1, it can be seen that the constant parameter curves are not spatially equidistant.

Two separate runs are made with different starting configurations and seeds for the random number generator. Once again, if the two runs yield approximately the same result, it gives confidence that the solution is indeed near the best. The SA1 parameters are the same as the first run of the second example in Chapter 5.

The results of the two runs are shown in Table 6.1 along with results of the two runs with fixed knots, from the previous chapter. There is a significant decrease in the objective

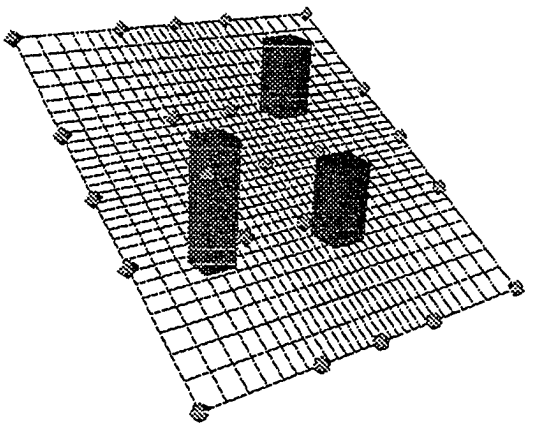


Figure 6.1 Initial configuration and location of knots

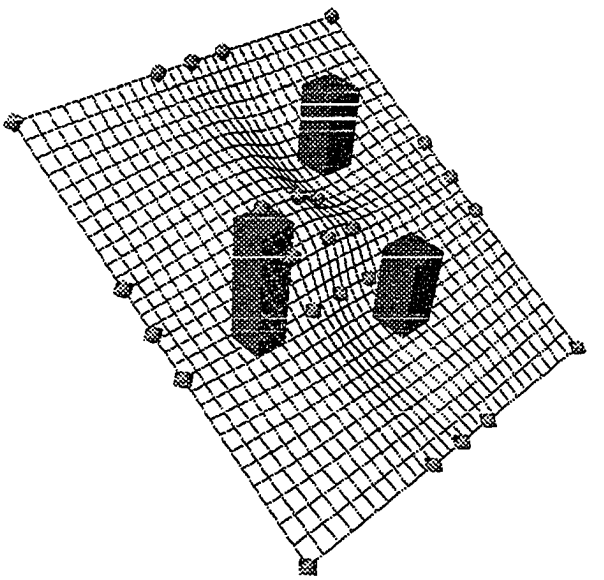


Figure 6.2 Final configuration and location of knots

function value with free knots. The parametric flow cost is influenced the most when the knots are freed. Figure 6.2 shows the final configuration of the surface with the knots. The spatial distribution of the constant parameter curves is fairly uniform in this case as opposed to the case with fixed knots (see Figure 5.5). The results of this example indicate that freeing of the knots can have a significant impact on the quality of the final surface. Surfaces with uniform spacing of the constant parameter curves are desirable for purposes of machining.

Table 6.1 Results (fixed and free knots)

Run	Arc cost	Para. flow cost	Total cost	Num. of function evaluations
Free knots 1	45.1997	51.3957	96.5954	582,368
Free knots 2	45.4428	50.9267	96.3695	582,249
Const. knots[a] 1	50.5350	149.4321	199.9671	522,646
Const. knots 2	51.8603	149.0112	200.8715	522,451

a. These results correspond to the runs of the previous chapter when the knots are fixed.

6.2. Surface with weights

This example demonstrates the generality of the surface synthesis technique. As mentioned in Chapter 2, rational B-splines are useful for the exact representation of natural quadrics. In fact, one of the most common uses of the rational form is for the representation of circular curves and cylindrical surfaces [Piegl and Tiller, 1987]. Figure 6.3 depicts a cylindrical NURBS surface in the vicinity of three polyhedral models. The surface is defined as quadratic in the radial direction and cubic in the axial direction. The control points in the radial direction are distributed as shown in Figure 6.4. There are eight control points in the axial direction, all equally spaced. The knot vector in the radial direction is:

$$U = \{0, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, 1, 1, 1\} \tag{6.1}$$

while in the axial direction it is:

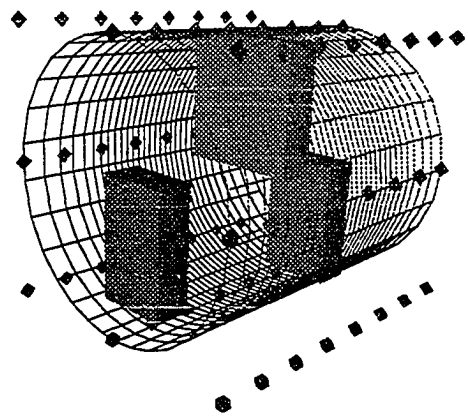


Figure 6.3 Cylindrical surface, initial configuration.

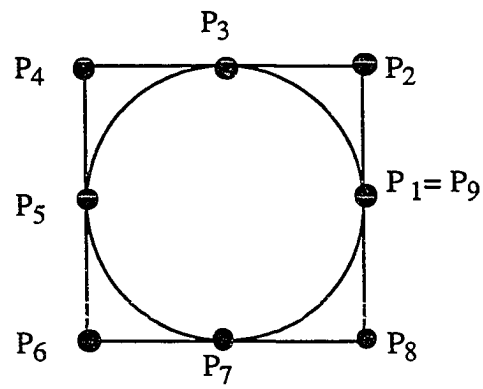


Figure 6.4 Control point distribution for a circle

$$V = \{0, 0, 0, 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, 1, 1, 1\} \tag{6.2}$$

and the weights in the radial direction are $W = \{1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1\}$. The control point net at both ends of the cylindrical surface is specified as fixed in all coordinate directions while the interior control points are fixed in the axial direction only. Thus, this problem has a

total of 96 degrees of freedom.

The surface is evaluated at 30 points in the u and v parameter directions, i.e., $M = N = 30$. Four constraints are applied to the design environment: the proximity constraint, arc length constraint, orthogonality constraint and parametric flow constraints (Chapter 3). The values used for cost penalties are $K_1 = 50.0$, $K_2 = 500.0$, $K_3 = 0.0$, $K_5 = 1.0$, $K_6 = 0.5$ and $K_7 = 5.0$. The arc length constraint is modified so as to be active only in the radial direction and not in the axial direction. If applied to the axial direction, the arc length constraint would have the effect of steering the design towards a cylinder because straight parameter curves would yield the smallest arc length.

Two runs with different initial conditions and random number seeds are made. The final configuration of the surface and the location of the control points are shown in Figure 6.5. The results of the two runs are shown in Table 6.2. The final costs of the two runs differ

Table 6.2 Results for the cylindrical model

Run	arc cost	ortho cost	para. cost	total cost	fun. evals.
1	233.659	52.589	286.895	573.141	678,324
2	230.703	53.798	286.801	571.302	686,728

by only 0.3% which is very good considering the use of relaxed parameters for the SA1 runs. On close inspection of the final control point locations of the two runs, they seem to take on identical locations.

6.3. Car model with symmetry constraints

This example was chosen to demonstrate one of the potential applications of this surface synthesis technique. Figure 6.4 shows the initial configuration of the surface in the vicinity of three (contacting) obstacles which are intended to represent the interior compartment of

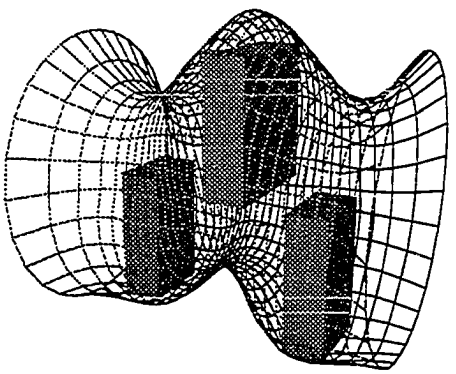


Figure 6.5 Cylindrical model, final configuration

a toy model automobile. Only half the car surface is synthesized due to symmetry. The surface is a non-rational bi-cubic B-spline with $12X8$ control points. The coordinate system is defined such that the x-axis is in the axial direction, the y-axis in the lateral direction and the z-axis ground-up. The control points at the lower edge of the surface are fixed in the z-direction while those at the upper edge are fixed in the y-direction. The control points at the front edge and the rear edge are fixed in the x-direction. This problem has 248 degrees of freedom. The surface is evaluated at 30 points in the u and 20 in the v parameter directions, i.e., $M = 30$ and $N = 20$.

Five constraints are applied to the design environment: the proximity constraint, arc length constraint, orthogonality constraint, parametric flow constraint and a symmetry constraint (see Chapter 3). The arc length constraint is enforced only the v-parameter direction as application in the u direction would cause the constant v parameter lines to straighten out. The tangent vectors at the plane of symmetry (xz-plane) are constrained to have a zero slope

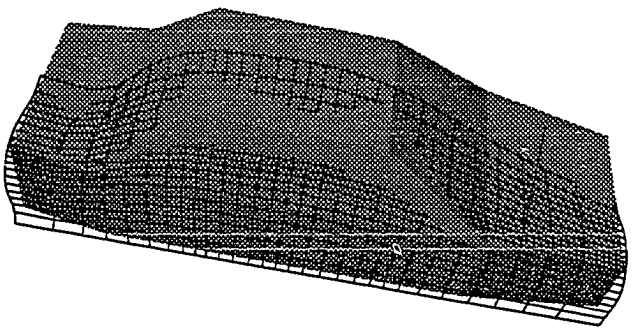


Figure 6.6 Car model, initial configuration

in the z-direction. The proximity constraint is enforced as follows:

$$C_{ij} = \begin{cases} K_1 & \text{if} \quad D_{ij} \le 0 \\ K_2(\delta - D_{ij}) & \text{if} \quad 0 < D_{ij} \le \delta \\ 0 & \text{if} \quad \delta < D_{ij} \le 3\delta \\ K_2(D_{ij} - \delta) & \text{if} \quad 3\delta < D_{ij} \le 4\delta \\ K_3 & \text{if} \quad D_{ij} > 4\delta \end{cases} \tag{6.3}$$

The values used for cost penalties are $K_1 = 50.0$, $K_2 = 500.0$, $K_3 = 500.0$, $K_5 = 1.0$, $K_6 = 1.0$, $K_7 = 5.0$ and $K_{10} = 10.0$.

In this case only one run is made. The results of this run are shown in Table 6.3. The final configuration is shown in Figure 6.7.

Table 6.3 Result for the car model

prox. cost	arc cost	ortho cost	para. cost	sym. cost	total cost	fun. evals.
492.539	66.445	66.944	531.372	0.000	1157.300	1,831,415

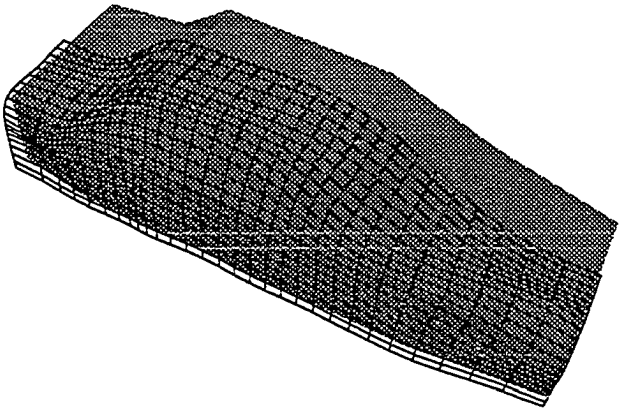


Figure 6.7 Car model, final surface configuration

CHAPTER 7. CONCLUSIONS AND FUTURE RESEARCH

7.1. Conclusions

In this dissertation a new technique for automatic, constraint-based synthesis of sculptured surface models is presented. A distance constraint is formulated to enforce spatial satisfaction and several auxiliary constraints, derived from analytic surface properties, have been developed to improve the quality and manufacturability of the surface, and exploit part symmetry. For a given configuration of the surface, dissatisfaction of any of the active constraints results in a penalty. The penalties due to different constraints is summed to yield the total cost. The cost is a reflection of the quality of the surface. The goal is to minimize the cost which is tantamount to optimization.

Initially it is not known whether the problem is unimodal or multimodal and therefore, which optimization method would be best. Consequently, a set of five optimization techniques (Nelder-Mead simplex, conjugate gradient, quasi-Newton, and two versions of simulated annealing) are used to solve two example problems. The results of the experiments reveal that the surface synthesis problem is multimodal in nature. This rules out the use of techniques such as Nelder-Mead, conjugate gradient and quasi-Newton because they tend to converge to locally good solutions. One version of simulated annealing, ASA, also tends to converge to locally good solutions. The possible reasons are discussed in Chapter 5. The other simulated annealing algorithm, SA1, is the only method that yields good solutions to both problems in Chapter 5. Furthermore, SA1 addresses the concerns about robustness and selection of parameters such as initial temperature, step length and choice of cooling schedule expressed by Oliver and Theruvakattil [1993b]. Consequently, SA1 is the choice for automatic generation

of sculptured surface models. A number of additional examples using SA1 are presented in Chapter 6. Each of the additional examples introduces a new feature.

This new technique is expected to form the basis of a completely automated system for the synthesis of sculptured surface models. Such a tool is not envisioned to provide a highly precise, production-ready surface model, but rather an initial model that globally satisfies many known constraints. This will provide surface designers with a rapid analytical prototyping facility, which would foster experimentation with novel configurations and enhance creativity.

7.2. Future research

The successful implementation of the general surface synthesis method has opened many potential avenues for further research. An important area of future research is the characterization of additional design and manufacturability constraints in terms of analytical surface properties. For example, earlier work in curve synthesis demonstrated the incorporation of constraints based on curvature and parametric distribution, and described their relationship to machinability [Malhotra et al., 1992]. These constraints can be generalized for application to surfacing. In addition to obstacle proximity, other surface functionality constraints, based perhaps on hydrodynamic, aerodynamic or even kinematic performance indices could be developed.

To make this new technique more accessible as a general design tool, further research is necessary on methods to define and interact with the design environment. In particular, methods are required to facilitate the interactive specification of the design environment, including the spatial location of obstacles and surfaces, the number and distribution of various surface topologies (i.e., collections of rectangular and n-sided topologies), and the specification and maintenance of various geometric and parametric constraints. Techniques are needed to explore the feasibility of automatically generating a problem configuration given minimal

user input and direction.

Research is necessary in the general area of computational efficiency. The computational effort required for even the simplest problem is prohibitively high. Therefore, any improvements in computational efficiency would certainly enhance the viability of this surface synthesis methodology. Several aspects of this methodology are suited to parallelization. For example, the evaluation of the surface, which is necessary before the cost can be computed, can be done in parallel. The part of the program that computes the distance from a point on the surface to the polyhedral models can be implemented in parallel. The cost due to different constraints could be computed in parallel. Another potential area of research is that of a parallel simulated annealing algorithm.

Preliminary results indicate that this technique has potential applications as a conceptual design tool supporting concurrent product and process development methods.

BIBLIOGRAPHY

Andersson, E., Andersson, R., Boman, M., Elmroth, T., Dalhberg, B. and Johansson, B., 1988, "Automatic Construction of Surfaces with Prescribed Shape," *Computer Aided Design*, Vol. 20, No. 6, pp. 317-324.

Bennett, J.A. and Botkin, M.E., (eds.), 1986, *The Optimum Shape*, Plenum Press.

Blais, G. and Levine, M.D., 1993, "Registering Multiview Range Data to Create 3D Computer Objects," *TR-CIM-93-16*, Centre for Intelligent Machines, McGill University, Montreal, Quebec, Canada

Bohachevsky, I.O., Johnson, M.E. and Stein, M.L., 1986, "Generalized Simulated Annealing for Function Optimization," *Technometrics*, Vol. 28, No. 3, pp. 209-217

Bohm, W., Farin, G. and Kahmann, J., 1984, "A Survey of Curve and Surface Methods in CAGD," *Computer Aided Geometric Design*, Vol. 1, pp. 1-60.

Casale, M.S. and Bobrow, J.E., 1988, "A Set Operation Algorithm for Sculptured Solids Modeled with Trimmed Patches," *Computer Aided Geometric Design*, Vol. 6, pp. 235-247.

Celniker, G. and Gossard, D., 1989, "Energy-Based Models for Free-Form Surface Shape Design," *ASME Advances in Design Automation, B. Ravani, ed.*, DE-Vol. 19-1, pp. 107-112.

Celniker, G. and Gossard, D., 1991, "Deformable Curve and Surface Finite Elements for Free-Form Shape Design," ACM Computer Graphics, Vol. 25, No. 4, pp. 257-266.

Cerny, V., 1985, "Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm," *Journal of Optimization Theory and Applications*, Vol. 45, pp. 41-51

Cobb, E.S., 1984, "Design of Sculptured Surfaces using the B-Spline Representation," *Doctoral Dissertation*, University of Utah, Salt Lake City.

Cohen, E., Lyche, T. and Riesenfeld, R.F., 1980, "Discrete B-splines and Subdivision Techniques in Computer Aided Geometric Design," *Computer Graphics and Image Processing*, Vol. 14, No. 2, pp. 87-111.

Corana, A., Marchesi, M., Martini, C. and Ridella, S., 1987, "Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm," *ACM Transactions on Mathematical Software*, Vol. 13, No. 3, pp. 262-280.

Cox, M., 1972, "The numerical Evaluation of B-splines," *Journal of the Institute of Mathematics Applications*, Vol. 10, pp. 134-149.

de Boor, C., 1972, "On Calculating with B-splines," *Journal of Approximation Theory*, Vol. 6, pp. 50-62.

Dekkers, A. and Aarts, E., 1991, "Global Optimization and Simulated Annealing," *Mathematical Programming*, Vol. 50, pp. 367-393

Devadas, S. and Newton, A.R., 1987, "Topological Optimization of Multiple-Level Array Logic," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6, No. 6, pp. 915-941

Dixon, J.R., Simmons, M.K. and Cohen, P.R., 1984, "An Architecture for Applying Artificial Intelligence to Design," *Proceedings of 21st ACM/IEEE Design Automation Conference*, Albuquerque, NM.

Elperin, T., 1988, "Monte Carlo Structural Optimization in Discrete Variables with Annealing Algorithm," *International Journal for Numerical Methods in Engineering*, Vol. 26, pp. 815-821.

Farin, G., 1993, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego.

Faux, I.D. and Pratt, M.J., 1979, *Computational Geometry for Design and Manufacture*, Ellis Horwood Ltd., Chichester.

Ferguson, D.R., 1986, "Construction of Curves and Surfaces Using Numerical Optimization Techniques," *Computer Aided Design*, Vol. 18, No. 1, pp. 15-19

Ferguson, D.R., Frank, P.D. and Jones, A.K., 1988, "Surface Shape Control Using Constrained Optimization on the B-spline Representation," *Computer Aided Geometric Design*, Vol. 5, pp. 87-103.

Foley, J.D., van Dam, A., Feiner, S.K. and Hughes, J.F., 1992, *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, MA.

Goffe, W.L., Ferrier, G.D. and Roger, J., 1994, "Global Optimization of Statistical Functions with Simulated Annealing," *Journal of Econometrics*, Vol. 60, No. 1/2, pp. 65-100

Goffe, W.L., 1994, "Simulated Annealing," [*netlib.ornl.gov: opt/simann.f*]

Haddock, J. and Mittenthal, J., 1992, "Simulation Optimization using Simulated Annealing," *Computers in Industrial Engineering*, Vol. 22, No. 4, pp. 387-395.

Haftka, R.T. and Gurdal, Z., 1993, *Elements of Structural Optimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Holland, J.H., 1973, "Genetic Algorithms and the Optimal Allocation of Trials," *SIAM Journal of Computing*, Vol. 2, No. 2, pp. 88-105

Imam, H.M., 1982, "Three-Dimensional Shape Optimization," *International Journal for Numerical Methods in Engineering*, Vol. 18, pp. 661-673.

Ingber, L., 1989, "Very fast Simulated Re-annealing," *Mathematical and Computational Modelling*, Vol. 12, No. 8, pp 967-973

Ingber, L., 1993, "Simulated Annealing: Practice versus Theory," *Mathematical and Computational Modelling*, Vol. 18, To be published.

Ingber, L., 1994, "Adaptive Simulated Annealing (ASA)," [*ftp.caltech.alumni.edu: /pub/ingber/ASA-shar.Z*], Lester Ingber Research, McLean, VA (1994).

Jain, P. and Agogino, A.M., 1988, "Optimal Design of Mechanisms Using Simulated Annealing: Theory and Applications," *ASME Advances in Design Automation, S.S. Rao, ed.*, DE-Vol. 14, pp.233-240.

Jensen, T.W., Petersen, C.S. and Watkins, M.A., 1991, "Practical curves and surfaces for a geometric modeler," *Computer Aided Geometric Design*, Vol. 8. No. 5, pp. 1-6.

Karmarkar, N., 1984, "A New Polynomial Time Algorithm for Linear Programming and Applications," *Proceedings 16th ACM Symposium on Theory of Computing*, New York, NY.

Kaufmann, E. and Klass, R., 1988, "Smoothing Surfaces Using Reflection Lines for Families of Splines," *Computer Aided Design*, Vol. 20, No. 6, pp. 312-316.

Kim, K. and Biegel, J.E., 1988, "A Path Generation Method for Sculptured Surface Manufacture," *Computers in Industrial Engineering*, Vol. 14, No. 2, pp. 95-101.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., 1983, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671-680.

Kosters, M., 1991, "Curvature-Dependent Parametrization of Curves and Surfaces," *Computer-Aided Design*, Vol. 23, No. 8, pp. 569-578.

Lott, N.J. and Pullin, D.I., 1988, "Method for Fairing B-spline Surfaces," *Computer Aided Design*, Vol. 20, No. 10, pp. 597-604.

Malhotra, A., Oliver, J. H., and Tu, W., 1991, "Synthesis of Spatially and Intrinsically Constrained Curves," *ASME Advances in Design Automation, DE-Vol. 32-1, G. Gabriele,ed.*, pp. 145-155

Metropolis, N., Rosenbluth, A., Rosenbluth, M. and Teller, A., 1953, "Equations of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, Vol. 21, pp. 1087-1091.

Michalewicz, Z. and Janikow, C.Z., 1991, "Genetic algorithms for numerical computation," *Statistics and Computing*, Vol. 1, pp 75-91

Mortenson, M. E., 1985, *Geometric Modeling*, Wiley, New York.

Nair, N.K. and Oliver, J.H., 1993, "An Area Preserving Transformation Algorithm for Press Forming Blank Development," *ASME Advances in Design Automation*, DE-Vol. 65-1, B.J. Gilmore, ed., pp 515-523

Nevill, G.E. and Paul, G.H., 1987, "Knowledge-Based Spatial Reasoning for Designing Structural Configurations," *Proceedings of ASME Computers in Mechanical Engineering Conference*, New York, NY.

Nevins, J.L. and Whitney, D.E., 1989, *Concurrent Design of Products and Processes: a Strategy for the Next Generation in Manufacturing*, McGraw-Hill, New York.

Oliver, J.H. and Theruvakattil, P., 1993a, "Chapter 2: Automated Generation of Sculptured Surface Models," *Artificial Intelligence in Optimal Design and Manufacturing*, Z. Dong, ed., Prentice Hall Environmental and Intelligent Manufacturing Systems Series, M. Jamshidi, Series ed.

Oliver, J.H., and Theruvakattil, P., 1993b, "Sculptured Surface Model Synthesis based on Functional Design Constraints," *ASME Advances in Design Automation*, DE-Vol. 65-2, B.J. Gilmore, ed., pp 515-523.

Piegl, L. and Tiller, W., 1987, "Curve and Surface Constructions Using Rational B-splines," *Computer Aided Design*, Vol. 19, No. 9, pp. 485-498.

Piegl, L., 1989, "Modifying the Shape of Rational B-Splines. Part 2: Surfaces," *Computer Aided Design*, Vol.21, No. 9, pp. 538-546.

Piegl, L., 1991, "On NURBS: A Survey," *IEEE Computer Graphics and Applications*, Vol. 11, No. 1, pp. 55-71.

Platt, J. and Barr, A., 1988, "Constraint Methods for Flexible Models," *Computer Graphics (Proceedings of SIGGRAPH'88)*, Vol. 22, No. 4, pp. 279-288.

Poeschl, T., 1984, "Detecting Surface Irregularities Using Isophotes," *Computer Aided Geometry*, Vol. 1, pp. 163-168.

Powell, M.J.D., 1975, "Restart Procedures for the Conjugate Gradient Method," Mathematical Programming, Vol. 12, pp. 241-254

Reklaitis, G.V., Ravindran, A. and Ragsdell, K.M., 1983, *Engineering Optimization, Methods and Applications*, John Wiley and Sons, New York.

Riesenfeld, R., 1973, "Applications of B-spline approximation to geometric problems of computer aided design," *Doctoral dissertation*, Syracuse University, Syracuse, New York, USA.

Romeo, F. and Sangiovanni-Vincentelli, A., 1991, "A Theoretical Framework for Simulated Annealing," *Algorithmica*, Vol. 6, pp 302-345.

Rutenbar, R.A., 1989, "Simulated Annealing Algorithms: An Overview," *IEEE Circuits and Devices*, January, pp. 19-26.

Saia, A., Bloor, M.S. and dePennington, A., 1989, "On the Integration of Parametric Polynomial Surface Representations into CSG Based Solid Modeling," *ASME Advances in Design Automation, B. Ravani, ed.*, DE-Vol. 19-1, pp. 121-127.

Sandgren, E. and Venkataraman, S., 1989, "Robot Path Planning via Simulated Annealing: A Near Real Time Approach," *ASME Advances in Design Automation, B. Ravani, ed.*, DE-Vol. 19-1, pp. 345-351.

Shah, J.J., 1988, "Synthesis of Initial Form for Structural Shape Optimization," *Transactions of the ASME, Journal of Vibration, Acoustics, Stress, and Reliability in Design*, Vol. 110, pp. 564-570.

Szeliski, R. and Tonnesen, D., 1992, "Surface Modeling with Oriented Particle Systems," *Computer Graphics (Proceedings of SIGGRAPH '92)*, Vol. 26, No. 2, pp. 185-194

Terzopoulos, D., Platt, J., Barr, A. and Fleischer, K., 1987, "Elastically Deformable Models," *Computer Graphics (Proceedings of SIGGRAPH'87)*, Vol. 21, No. 4, pp. 205-214.

Tiller, W., 1983, "Rational B-Splines for Curve and Surface Representation," *IEEE Journal of Computer Graphics and Applications*, Vol. 3, No. 10, pp. 61-69.

Tovey, M., 1989, "Computer-aided vehicle styling," *Computer Aided Design*, Vol. 21, No. 3, pp. 172-181.

Vanderbilt, D. and Louie, S.G., 1984, "A Monte Carlo Simulated Annealing Approach to Optimization over Continous Variables," *Journal of Computational Physics*, Vol. 56, pp 259-271.

van Laarhoven, P.J.M. and Aarts, E.H.L., 1987, *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, Dordrecht, Holland.

Watt, A. and Watt, M., 1992, *Advanced Animation and Rendering Techniques - Theory and Practice*, Addison-Wesley Publishing Company.

Weitzman, L., 1992, "Designer: A Knowledge-Based Graphic Design Assistant," Chapter 14, *Artificial Intelligence in Engineering Design*, Tong, C and Sriram, D. (eds.), Vol. 1, pp. 433-466.

Welch, R.V., and Dixon, J.R., 1989, "Extending the Iterative Redesign Model to Configuration Design: Sheet Metal Brackets as an Example," *ASME Design Theory and Methodology - DTM'89*, DE-Vol. 17, pp. 81-88.

Williams, C.J.K., 1987, "Use of Structural Analogy in Generation of Smooth Surfaces for Engineering Purposes," *Computer Aided Design*, Vol. 19, No. 6, pp. 310-322.

Woo, T.C. and Shin S.Y., 1985, "A Linear Time Algorithm for Triangulating a Point Visible Polygon," *ACM Transactions on Graphics*, Vol. 3, No. 2, pp. 60-70.

Wu, K. and Levine, M.D., 1993, "3-D Object Representation using Parametric Geons," *TR-CIM- 93-13*, Centre for Intelligent Machines, McGill University, Montreal, Quebec, Canada

Wysocki, D.A., Oliver, J.H. and Goodman, E.D., 1989, "Gouge Detection Algorithms for Sculptured Surface NC Generation," *ASME Computer-Aided Design and Manufacture of Cutting and Forming Tools*, PED-Vol. 40, pp. 39-44, (also, to appear, *Transactions of the ASME, Journal of Engineering for Industry*).