Designing Aesthetically Pleasing Freeform Surfaces in a Computer Environment

by

Evan P. Smyth

B.Sc., Mathematics, University of Notre Dame,1990
B.Arch., Architecture, University of Notre Dame, 1990
M.Des.S., Science and Technology, Harvard Graduate School of Design, 1991

Submitted to the Department of Architecture
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Architecture: Design and Computation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

Author .....
Department of Architecture
October 16, 2000

Certified by .....
William J. Mitchell
Professor of Architecture and Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Stanford Anderson
Chair, Committee on Graduate Studies
Head, Department of Architecture

Committee

Julie O'Brien Dorsey, Associate Professor of Architecture and Computer Science and Engineering

David Clair Gossard, Professor of Mechanical Engineering

Nicholas Marinos Patrikalakis, Kawasaki Professor of Engineering, Department of Ocean Engineering

Designing Aesthetically Pleasing Freeform Surfaces in a Computer Environment

by

Evan P. Smyth

Submitted to the Department of Architecture
on October 16, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Architecture: Design and Computation

Abstract

Statement: If computational tools are to be employed in the aesthetic design of freeform surfaces, these tools must better reflect the ways in which creative designers conceive of and develop such shapes.

In this thesis, I studied the design of aesthetically constrained freeform surfaces in architecture and industrial design, formulated a requirements list for a computational system that would aid in the creative design of such surfaces, and implemented a subset of the tools that would comprise such a system. This work documents the clay modeling process at BMW AG., Munich. The study of that process has led to a list of tools that would make freeform surface modeling possible in a computer environment. And finally, three tools from this system specification have been developed into a proof-of-concept system. Two of these tools are sweep modification tools and the third allows a user to modify a surface by sketching a shading pattern desired for the surface. The proof-of-concept tools were necessary in order to test the validity of the tools being presented and they have been used to create a number of example objects. The underlying surface representation is a variational expression which is minimized using the finite element method over an irregular triangulated mesh.

Thesis Supervisor: William J. Mitchell
Title: Professor of Architecture and Media Arts and Sciences

Acknowledgments

I want to thank all of the people who have played a role in bringing this work to its completion.

I thank my thesis advisor, Professor Bill Mitchell, for supporting me in this work from the very beginning. It was he who originally introduced me to many of the challenging surface modeling problems that would ultimately inspire me to undertake this work. Over time, he has continued to enthusiastically support and guide my work.

In addition, I thank my thesis committee for the interesting conversations we have had along the way as well as for their thorough review of drafts of the dissertation. Professors Julie Dorsey, David Gossard and Nicholas Patrikalakis have all greatly enriched the quality of this work.

I thank the people in the Styling Department at BMW AG., Munich, Germany, for allowing me to begin to appreciate the art of model making and encouraging me to develop that appreciation into a tangible design for a new system. The support I received from people there has been extraordinary and has, in many ways, helped to imbue this work with a depth it might not otherwise have had.

I thank the people at the office of Frank O. Gehry for giving me the original opportunity to develop a control model for the sculpture at Barcelona's Olympic Village which heightened my interest in this area.

I thank my father, Brian Smyth, for his helpful comments on a dissertation draft. I thank Renee Caso and Cynthia Wilkes, without whom coordinating the final stages of this work from Los Angeles would have been very difficult – if not impossible.

But perhaps most of all, I thank my wife, Janet Herold, who has been unwavering in her support and encouragement of me in this work despite the sacrifices in time that were required. Her patience and enthusiasm have been essential – to say nothing of her helpful readings of dissertation drafts.

Contents

List of Figures

Chapter 1

Introduction

As in other professions, the introduction of computational tools has had profound effects on the design professions. The enormous advantages gleaned from archiving colossal amounts of data efficiently, as well as maintaining the ability to modify design data later in the process while keeping others in the process apprised of any changes, have served to make computation an integral part of almost all design processes.

Architects have, for the most part, committed themselves to using computer-aided design (CAD) applications in their documentation work and, more recently, using them to enhance their client presentations as well. Engineering design has also been revolutionized by this technology perhaps because it is now possible to create "virtual" prototypes of designs (for testing purposes) instead of having to laboriously create physical prototypes. For instance, a great deal of automotive crash testing is now done using sophisticated Finite Element Methods (FEMs). While, physical crashes must ultimately be performed, FEMs have proven to be a very effective and reliable means to predict crash-worthiness. In effect, then, FEMs reduce the amount of time needed in the design process for costly production of prototypes.

Of course, there are many other examples that could be mentioned here, but it is interesting to note that very few (if any) of these applications deal with *aesthetically-constrained three-dimensional design.* Since this type of design is central to this research, it is worth defining it precisely at the onset of the discussion:

> **Aesthetically-constrained three-dimensional design:** The development of three-dimensional form using subjective criteria as a measure of quality. (eg. Architectural sketching, automotive styling).

Clearly, this is distinct from engineering design which tends to apply more objective (quantifiable) criteria to assess quality.[1]

Graphic design is, of course, aesthetically-constrained but it is obviously not (generally) three-dimensional. Graphic designers have been very quick to seize upon the benefits of computational tools; in fact, very little of this type of design is now done in any other medium.

[1] I will often refer to aesthetically-constrained three-dimensional design as *aesthetically-constrained design, aesthetic design, creative design* or even, simply, *design* when there is little chance of confusion.

The extent to which special effects in the entertainment industry are created using computers seems, at first, to contradict the assertion that there are few examples of computers being successfully applied in creative three-dimensional design. However, although it is true that many of the designs developed for special effects are fundamentally three-dimensional, it is also true, in the strictest sense, that such effects are judged in terms of how well they convey a three-dimensional idea in the two-dimensions of screen space. Thus, in many ways, this work is more like a very elaborate and complex graphic design than it is like three-dimensional design. In the entertainment realm, three dimensional design is best thought of as a means to an end. In contrast, to the industrial designer and architect, the final three-dimensional geometry is the end in itself.

Similarly, while computational tools are finding an increasing number of applications in three-dimensional creative design, these applications are usually those in which the designs themselves are relatively simple. Thus, the financial benefits of developing the designs in a computational medium already far outweigh the difficulties involved in the development. (Perhaps because the aesthetic demands are not so high, for instance).

Areas of design where the computer has been conspicuously absent include architectural design and automotive styling. These are the areas of primary concern in this research. Professionals in each discipline have different reasons for resisting the introduction of computational tools in their work.

Architects, who do not have high volume demands on their work, have little incentive to invest in expensive equipment before it is proven to be useful as a design tool. Since the computer has not proven itself as a valuable design tool in an architectural context, architects have tended to take a back-seat in the development process of new tools.

Automotive stylists enjoy the benefits of high volume and, as a consequence, have been more willing to experiment with these new technologies. Nevertheless, as a general rule, these designers have not adopted the tools for much more than detail work (i.e. wheel rims, door handles, etc.) and their presentations.

It is important to note that even though the work of the typical architectural firm – as well as the that of the typical automotive styling department – has generally been completed with the help of at least some computation, it is extremely rare that such help would have come in the early design phases: In other words, those phases in the design process where the conceptual design strokes are resolved. As unpleasant as the term is, this is often referred to as the ideation phase. Both architects and industrial designers generally rely heavily on sketches in this early stage. Once some promising ideas have been distilled from sketches, a designer will often make models to more fully explore the forms being created.

The reasons for the preference for these traditional design methods are fairly simple. By sketching, architectural and industrial designers can explore design spaces extremely rapidly. Thus, it is only after the designer has begun to identify some of the more promising ideas that she begins to give her conceptual design a more tangible form. After she chooses a particular design path to explore, she will make a scale model in order to verify that the sketched ideas can be converted into a three dimensional form. Such scale models also permit the presentation of ideas to others less comfortable with the sketch medium.

Our research here is aimed at examining a specific type of design which is a concern in each of these two design disciplines: namely freeform surface modeling. That is, after the designer has narrowed in on a design, she will need to begin to give that design three-

Figure 1-1: View of Günther Behnisch's 1972 Olympic Complex in Munich.

dimensional form. If that design involves freeform surfaces, subsequent model-making must obviously involve such surfaces as well. Automotive stylists generally do this work in plasticine clay. Examples of freeform surfaces in architecture are uncommon so it is more difficult to generalize about that field. And yet, architects are beginning to explore more complex freeform design aggressively, and this research should also be viewed as an early step toward clarifying some of the issues encountered when designing freeform surfaces.

In architecture, such freeform surface design is relatively uncommon. Most architectural work is dominated by planar surfaces. Even when, as in the case of Gunther Behnisch's 1972 Olympic complex (see Figure 1-1), for example, an architect uses curved surfaces in a design, these are very frequently simply cylindrical, parabolic, ellipsoidal, hyperbolic or minimal surfaces. This is not to say that there are not many challenging design problems which arise in such design (as well as in the design of planar surfaces), but this is not the type of surface design that this research addresses. Architectural surfaces which are relevant to this work include many of those created by Frank Gehry, such as his sculpture at the Olympic Village, Barcelona, 1992 (Figure 1-2) and Antonio Gaudí's Casa Milá, Barcelona, 1910 (Figure 1-3).

One way of expressing the commonality of these types of curved-surface designs is that the type of surfaces which are of interest here are those which *might well have been designed/modeled with clay* whether they actually were or not. Clearly, developing a model for the 1972 Olympic complex using clay would have been far from convenient. In that particular example, using soap-films would have been much more fruitful – in fact, this is how Behnisch developed the design with the help of engineer Frei Otto. Similarly, the form

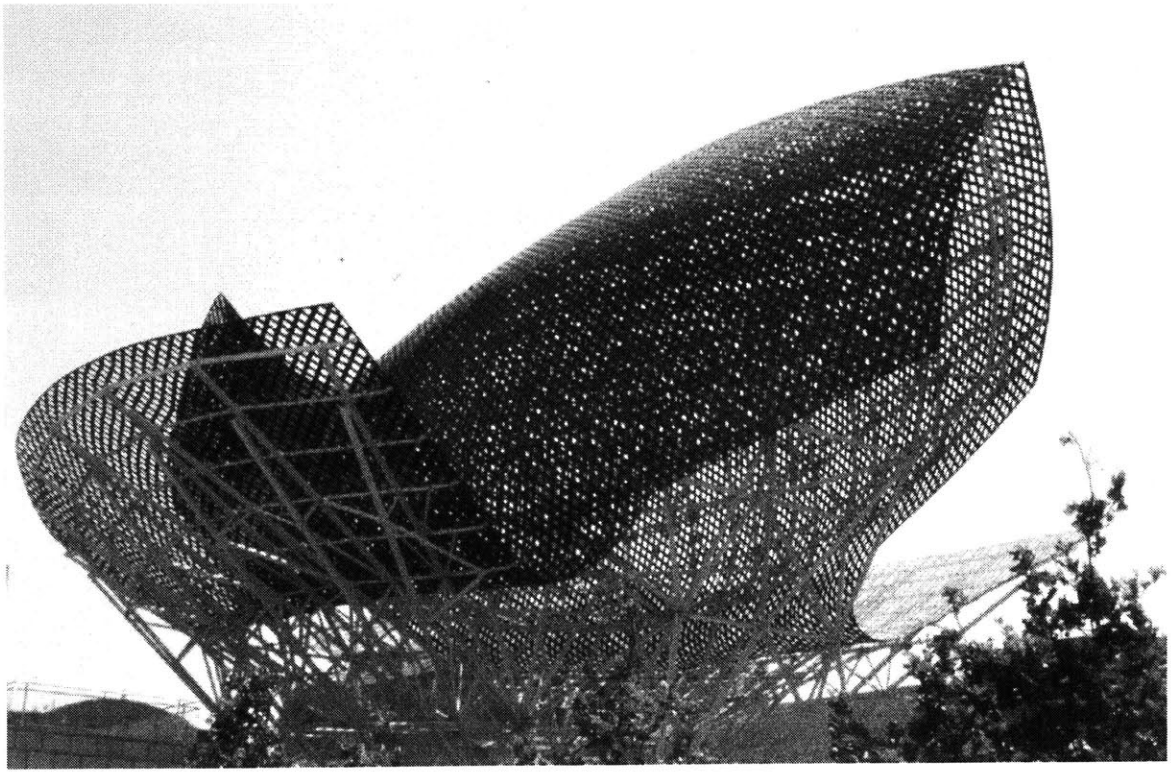


Figure 1-2: Frank O. Gehry's Olympic Village sculpture in Barcelona. 1992.

Figure 1-3: Antonio Gaudí's Casa Mila in Barcelona. 1910. Reproduced from [BL99]

of Gaudí's *Sagrada Familia* cathedral in Barcelona, which is composed of curved surfaces, was modeled using chains with weights attached so as to approximate catenary curves.

In concentrating on this particular and important type of freeform surface design, this research addresses how designers of these forms think about, communicate, and ultimately execute their work. In parallel, I will investigate why it is that these designers are unable to use the current computational tools. A proof-of-concept surface modeling application – based upon this particular type of surface design – is developed and implemented. In addition, this newly developed freeform surface design application is tested specific design contexts to determine its suitability to the types of design work being addressed and to identify areas of future research.

1.1 The Motivation

This thesis takes the position that we, as creative designers, must begin to define the extent to which our work is quantifiable and, perhaps just as important, to what degree it is not. In large measure, *computer aided geometric design* (CAGD) and CAD research has been driven by the quest to produce surfaces – or geometry in general – with certain

desirable quantifiable properties and to do so as rapidly as possible. Thus, traditional CAGD research and application approaches will likely not provide the necessary framework in which to implement a system based on the design metaphors described in this work.

In fact, traditional tools are not particularly appropriate for the aesthetic design process. To understand why this is true, it is useful to consider the typical media in which designers do this work. To the extent that architects and industrial designers develop the designs in a medium enabling communication, they typically sketch, draw or work with rough models. Their choice of medium reflects their interest in communicating their ideas very quickly and effectively – especially to others who are trained in related fields. This process, even if undertaken by a lone designer, can be thought of as a conversation. If it were to take a day to utter a sentence, it would not be a fruitful conversation. However, with tools such as pencil and paper, the designer can sketch and thereby resolve her ideas in an interactive way.

The critical idea here is the speed with which ideas are explored and rejected. Conventional computer tools do not lend themselves to such exploration and are thus wholly unsuited to the design development process – especially the early stages of that process. Consequently, *flexibility* is a vital ingredient in the design development stage. As a design is developed and refined, it begins to make more and more sense to spend additional time specifying properties of the design. Aspects may still change in the future, but the frequency and degree of changes generally decline as a healthy design process continues. It should be clear that in the course of this process, designers will generally become more and more willing to invest the time and energy into creating more and more accurate models of their work.

Up to now, from an aesthetic perspective, computational tools generally require a significant investment of time in resolving design issues before the designer begins to reap the benefits of the tools. Those benefits may appear to be great. For instance, a designer might be able to produce a photo-realistic rendering of a design proposal or synthetically place it in a wind-tunnel. However, while important in the overall design process, such benefits are marginal until a promising design emerges.

Flexibility is probably the single most important characteristic of a design process. Of course, a single word, such as "flexibility" always masks a host of other important considerations, but it is nevertheless true to say that most aesthetically-constrained design is characterized by a decidedly lateral process (particularly in the early stages), especially when compared to typical engineering design processes. For the designer, the importance of being able to effortlessly explore radically different avenues of thought is paramount. Any design system which does not support such lateral design processes will inevitably meet with resistance from designers. However, merely providing for a lateral design process is not enough either. It must also be possible to fully resolve the design as well.

Previous computer-aided design research has, in large measure, been motivated by the need to produce geometry with near-optimal quantifiable properties. For example, a surface is of a high quality in an aerodynamic sense if it has a sufficiently low coefficient of drag (*cd*). Nevertheless, that same surface may not be satisfactory from an aesthetic point of view. The engineer can justify using this drag coefficient in order to assess a surface because experience shows that a sufficiently low number corresponds to low fuel consumption or low wind noise, for example. The designer cannot take refuge in such a number and, instead,

makes assessments of the surface based upon her own taste, experience and also to some degree, upon quantitative feedback, perhaps even including some aerodynamic concerns.

The designer's work will be partially assessed in terms more like those involved in fine art. The surfaces must flow smoothly and have "nice" accelerations (Figure 1-4). The final

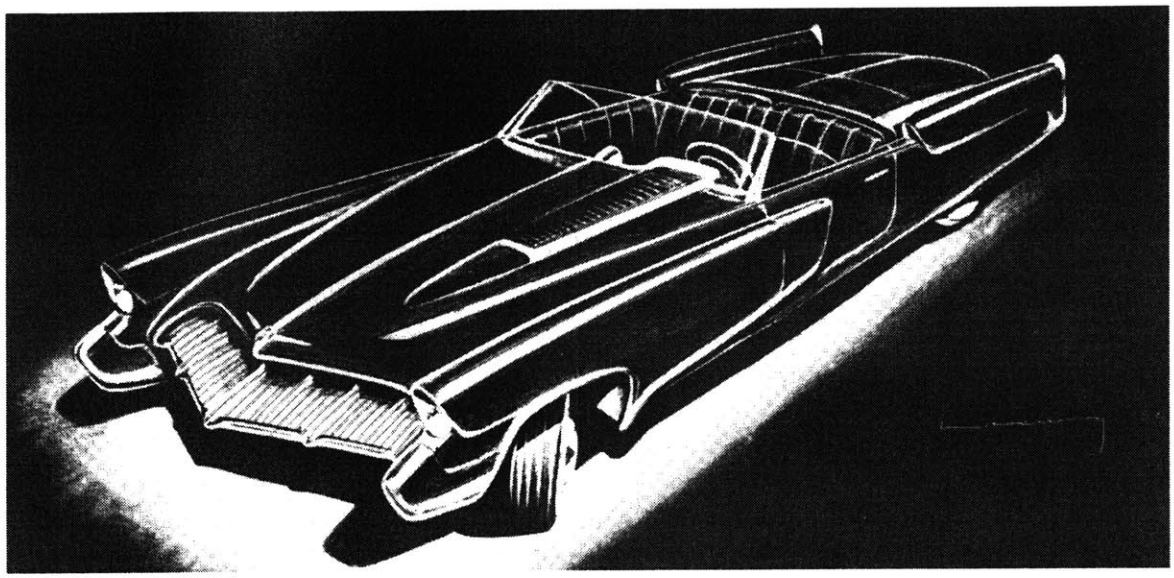


Figure 1-4: Raymond Loewy 1950 design embodying speed, large size and aggressive stance. Reproduced from [Loe79].

verdict on an industrial design (such as an automobile) will be delivered by the consumer and a similar reality reigns in other aesthetic design professions. Of course, no design professional has the luxury of being able to ignore all external non-aesthetic constraints (for instance, sheet metal can only be deformed in certain ways). Often, there will be various material constraints as well as those like the drag coefficient of the object.

An important observation at this point is that while these constraints or considerations must all be satisfied to some general degree, the designer enjoys a wide range of freedom in determining how best to satisfy those constraints and/or considerations. From the designer's point of view, these technical concerns are important but not determinative. In other words, they are secondary to the appearance or beauty of the object. Thus, the approach of attempting to define the problem as a mathematical one is necessarily doomed to fail.

Auto-stylist Henrik Fisker, then at BMW, Munich, and now president of DesignWorks, USA, observed to me that he saw no reason to fear the introduction and application of computers in his field because the computer would only become a serious rival medium when the designs created on them are the equal of those created in traditional ways. As for this designer, when that day comes, he too, would be willing to consider this new medium in his own work – but not much sooner. I think this pragmatic approach reflects more the rule than the exception within the design professions despite the fact that developers of many computational systems generally presume that it is the designers who are deficient and not their software.

This thesis can be viewed as an early attempt to formulate and implement a creative freeform surface design system based on the idea of providing maximal flexibility to the designer rather than providing a means to reach numerical targets. Such targets have little or nothing to contribute to the creative design process.

1.2 Surface Representation

The choice of surface representation will be treated in more detail in Chapter 5. It is useful, however, to briefly review some of the major factors at this time.

A significant requirement is that the resulting surface should not need to be stitched. In other words, we would like inter-surface boundary conditions to either be satisfied automatically or to not exist at all.

Another important requirement is that the representation not require support on a regular grid. Most currently used surface representations are supported on a rectangular grid and are thus very poorly suited to the sorts of localized modifications that are required of a good surfacing system.

The final important requirement is that surfaces created in this system should be smooth by some measure.

The surface representation used in this research is closely related to the ShapeWright (Shape "W" Right) representation presented by George Celniker in his doctoral dissertation [Cel90]. This representation is based upon a non-regular triangular mesh with degrees of freedom at the mesh vertices and mid-edge locations. Using finite element methods, a specialized variational functional is minimized over the mesh which in turn produces a surface. This surface is optimal in the sense that it minimizes a weighted sum of area and bending terms in the variational formulation. The relative weighting of terms in this variational formulation can be configured by the user, although, in practice, such user selection has proven extremely difficult.

Clearly, this representation should obviate the need for excessive stitching. Because meshes can be irregular, far fewer patches are required and surface derivatives at mesh vertices are explicitly represented in the system as degrees of freedom.

The question of smoothness is of great import and it will be discussed later. Suffice it to say that the representation can support arbitrary levels of smoothness in the mathematical sense. In this specific implementation, C^1 surfaces are guaranteed although Celniker makes a convincing argument that, although the surfaces are not C^2, they do approach that level of continuity in an approximate way – due to the fact that the variational formulation tends to distribute curvature of the surface, thus reducing the likelihood of sharp changes. It should also be noted that fully $C2$ and higher triangular finite element interpolants do exist and, with more time, could also be implemented. Of course, they become computationally more weighty.

1.3 Tools Developed in the Research

As part of this research, new tools have been developed which show great promise as tools which more closely relate to the ways in which designers actually work. This is not to

say that the only successful approach to creating such tools will come from mimicking current work methods. Yet, it does seem to be a logical place to start. Although the tools developed here owe at least some of their formulation to existing design techniques, in many respects, even these early attempts go beyond the design metaphors suggested by existing work methods.

There are three distinct tools developed in this research. Two different sweep modifier tools are developed as well as a tool based upon a class of machine vision techniques that has come to be known as *Shape from Shading* [Hor86, HB89].

1.3.1 Sweep Modifier Tools

These tools are primarily the result of investigations into contemporary industrial design techniques. Specifically, as a result of observations in an automotive styling studio, it became clear that having tools that were at least related to the process of sculpting three-dimensional models in clay would be invaluable in a computational system developed to aid aesthetic design of freeform surfaces.

Conceptually, these tools are also related to general sweep tools as described by Sabine Coquillart [Coq87]. Currently available sweep tools are based upon her original work and may be said to aid in the generation and modification of *entire* surfaces. The tools developed in this work are developed for the editing of preexisting surfaces. More precisely, they are well-suited for the editing of sub-regions of preexisting surfaces.

Each of the two sweep modifier tools have been implemented in two versions. The first, and simpler version of each literally modifies the underlying surface in a way determined by a sweep surface temporarily created by the user. The second, and far more powerful version, actually replaces a portion of the mesh controlling the underlying surface so that it more accurately captures the swept surface temporarily created by the user. It would be difficult, if not impossible, to implement the submesh-replacing versions of the these tools using currently popular surface representations.

Section Sweep Tool

The Section Sweep Tool is very well suited to precision work. Given an input curve on the surface to be modified, the user places section planes at arbitrary locations along the curve. Within each of these section planes, a curve is used to define what the cross-section of the surface is to be in the current section plane. In this way the user effectively and easily defines the shape of the surface.

In the simpler version of this tool (in which only mesh modification is performed), it should be clear that if the underlying mesh is not sufficiently dense, the resulting surface may well fail to capture a great deal of the detail that the tool's user may define using the sections.

The second version addresses this problem squarely. The simpler version of the tool determines the set of points on the existing mesh that need to be modified and literally moves them along the surface's normal direction (at each point) until they intersect with the temporarily created sweep surface. Thus, in the limit, if no mesh vertices happened to project onto the temporary surface, then the underlying surface would not be modified at all. In the second version, mesh elements that need to be modified are identified and

tagged. Next a new mesh is created that corresponds to the temporary swept surface and this new mesh is merged into the the underlying mesh in place of the tagged elements. The process by which this is done is surprisingly complex and will be described in more detail herein. However, the important point is that the resulting mesh captures the user-described shape accurately.

Dragging Sweep Tool

The Dragging Sweep Tool is much more intuitive than the Section Sweep Tool. A significant amount of future work remains for this tool, as its implementation seems very promising. It seems very likely that this tool could prove extremely effective in a virtual environment, especially in conjunction with customized hardware.

When using this tool, the user again begins with a path curve on the surface to be modified. Upon the user's command, the tool begins to travel along the path curve in the curve's parametric direction. This is accomplished by moving a section plane along the curve at some predefined velocity. The interesting aspect of this tool is that as the section plane is dragged along the curve, the user controls the shape of the curve in that plane by pressing specific keys on the keyboard. In fact, the curve is modeled as a single span cubic spline and four keys are assigned to each hand. Depending on how many keys are pressed the curve is bent appropriately. Clearly, with only four keys to each hand, the tool has only a very small number of sample points and hence, it has a small number of possible configurations. However, as a basic interface, it is satisfying and the simple addition of a dials box or some other continuously sampling interface would completely address that shortcoming.

1.3.2 Shape Shading Tool

The final tool developed in this work takes advantage of a shape from shading algorithm developed by Bichsel and Pentland [BP92]. The essence of this tool is the realization that it will often be convenient to actually sketch a surface shape (especially for a user talented at sketching) given a means to produce a three-dimensional form corresponding to the form implied by the sketch.

1.4 Organization

The next chapter details the results of my investigation into automotive styling along with some commentary on how future architectural design of freeform surfaces relates to such industrial design work. Chapter 3 covers the fundamental theory underlying the field of CAGD and Finite Elements Methods (FEMs) as they relate to this research. Chapter 4 details previous research relating to the problems encountered in my research. Chapter 5 describes the implementation produced in this research. Chapter 6 provides a number of example surfaces using the proof-of-concept system described in Chapter 5. Finally, in Chapter 7, conclusions are drawn and possible future work is mapped out.

Chapter 2

Freeform Surface Design Metaphors

In order to develop a model of a system which will allow a designer to comfortably use it to develop freeform surface designs, it is reasonable to begin by investigating how designers currently achieve this difficult task.

This research concerns itself with the development of computational tools to aid in the development and exploration of three-dimensional freeform shapes. The hope is that by formalizing the design process, this work will advance the development of design tools by attempting to address these formalisms in new ways.

The significance of these observations is that although a goal of this work is to establish a framework from which architects can approach such freeform surface design, it must also be realized that architects themselves have done very little work with freeform surfaces as a general matter. To the extent that they have, it has often been at great expense and with great difficulty. Nevertheless, there is a growing interest in such forms in architectural circles.

2.1 Architectural Freeform Surface Design

Architectural freeform surface design is more of an exception than a rule. Historically, architects have naturally gravitated toward producing designs based upon horizontal and vertical planes – for reasons both practical and of tradition. Horizontal planes make a great deal of sense as it is generally more appealing to walk on horizontal surfaces. In the situation where a multi-story structure is being designed, horizontal floors naturally encourage having horizontal ceilings. It should be noted that this does not, in and of itself, encourage the use of horizontal roofs. Horizontal roofs are with us primarily for reasons of cost and style. The use of vertical planes is most probably rooted in the fact that it is naturally easier to support a weight if you stand directly under it than if you hold it in your hand with your arm stretched out.

Surely the most prominent contemporary architect to explore curved surfaces in his work is Frank O. Gehry [LM95, Geh95b, Geh95a, Sch93]. While Frank Gehry has worked with truly freeform surfaces – in his Barcelona Olympic Village sculpture for example (see

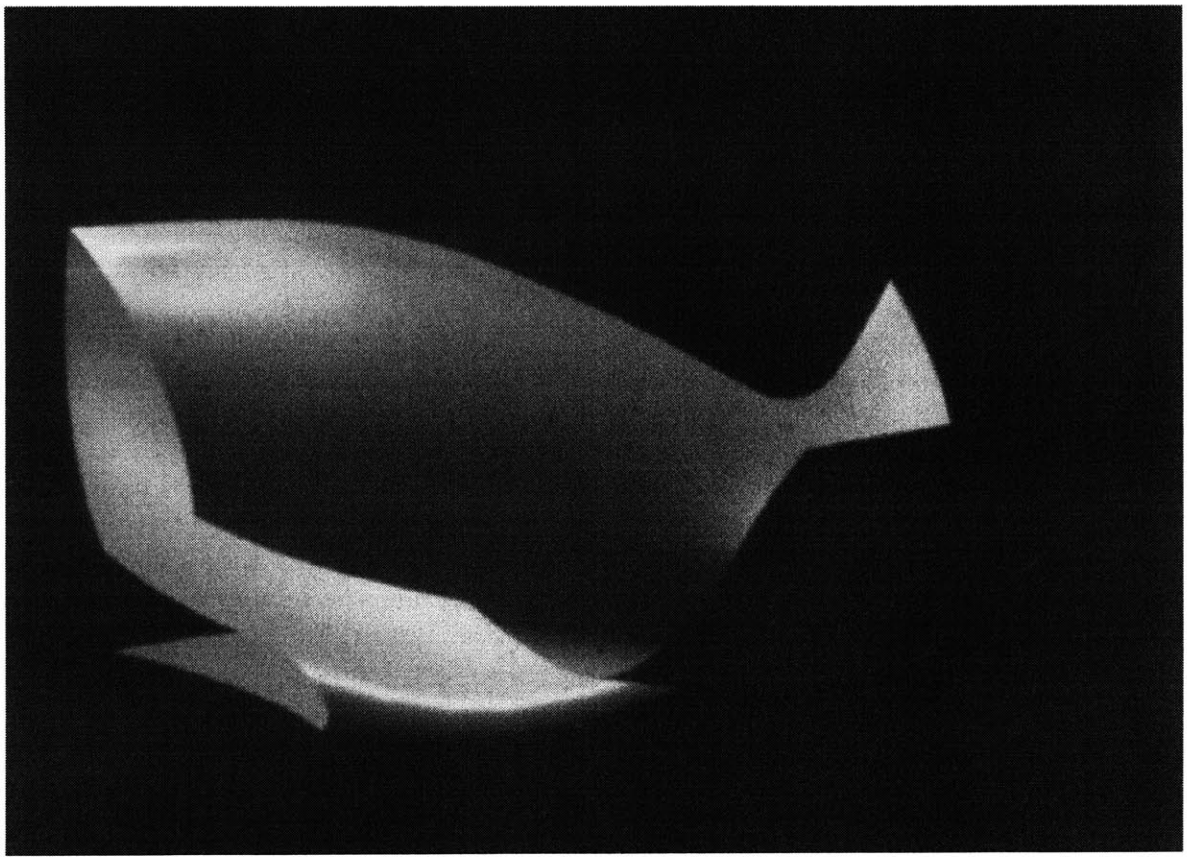


Figure 2-1: Control Model for Olympic Village sculpture I produced in 1991.

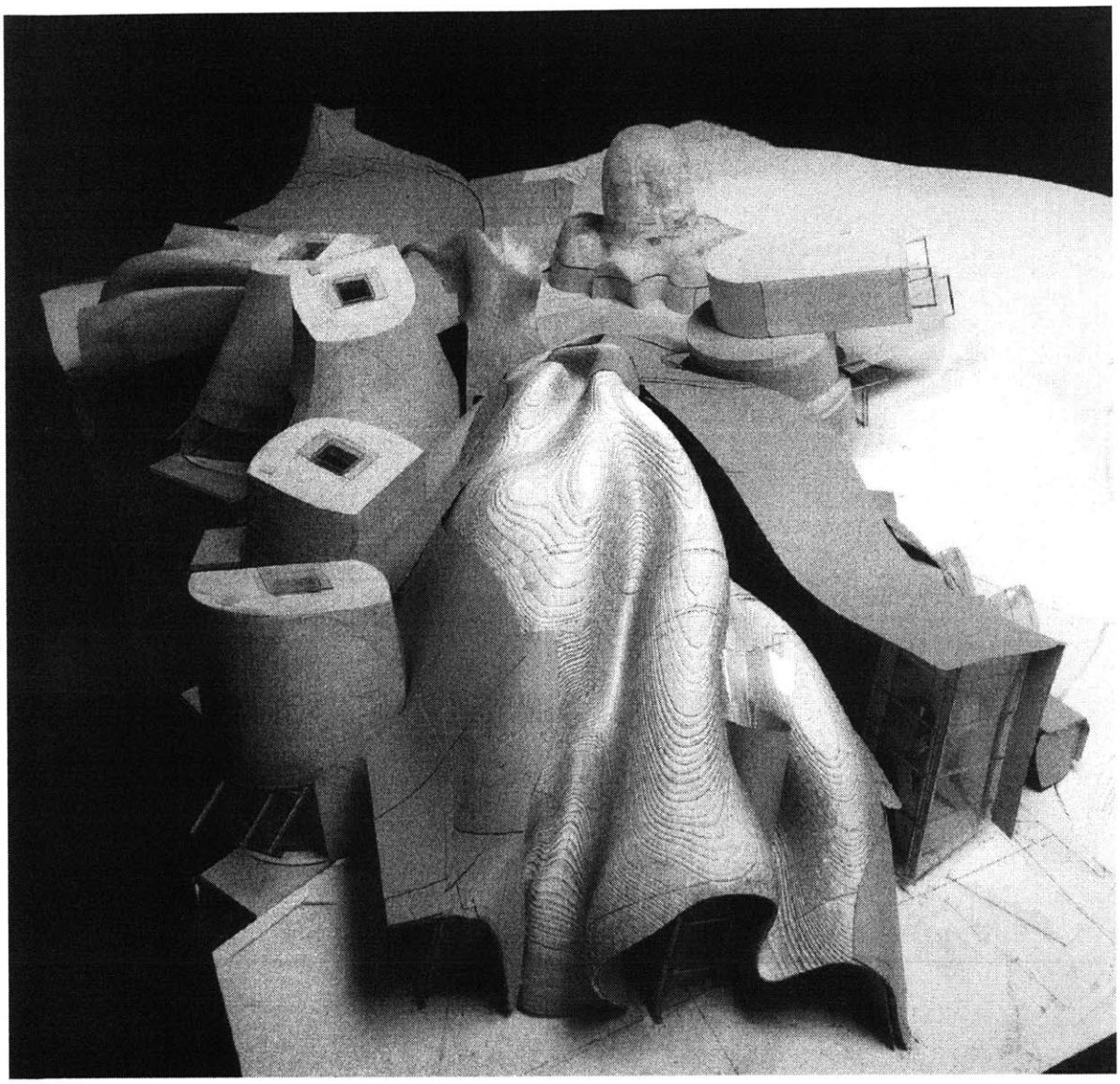


Figure 2-2: Lewis Residence. Model. 1989-1995. Reproduced from [LM95].

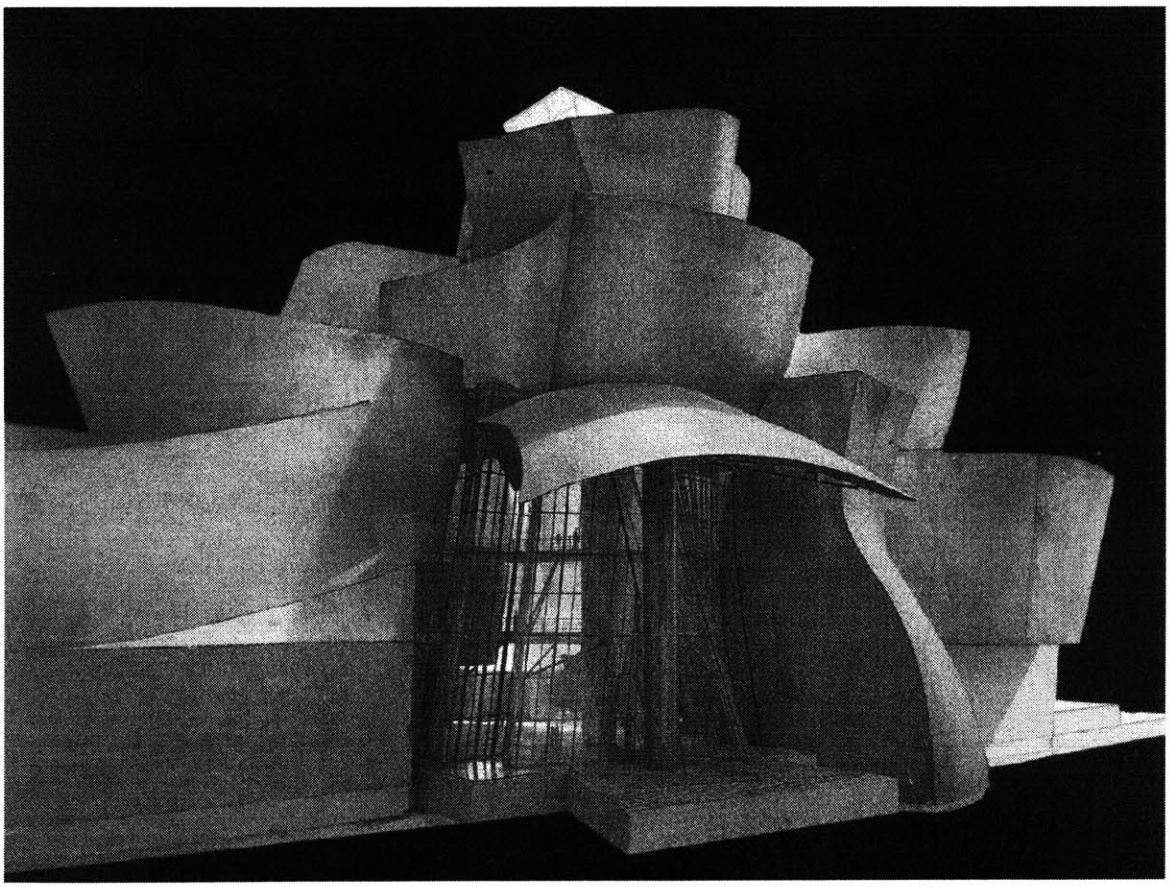


Figure 2-3: Guggenheim Museum, Bilbao, Spain. 1991-1997. Reproduced from [LM95].

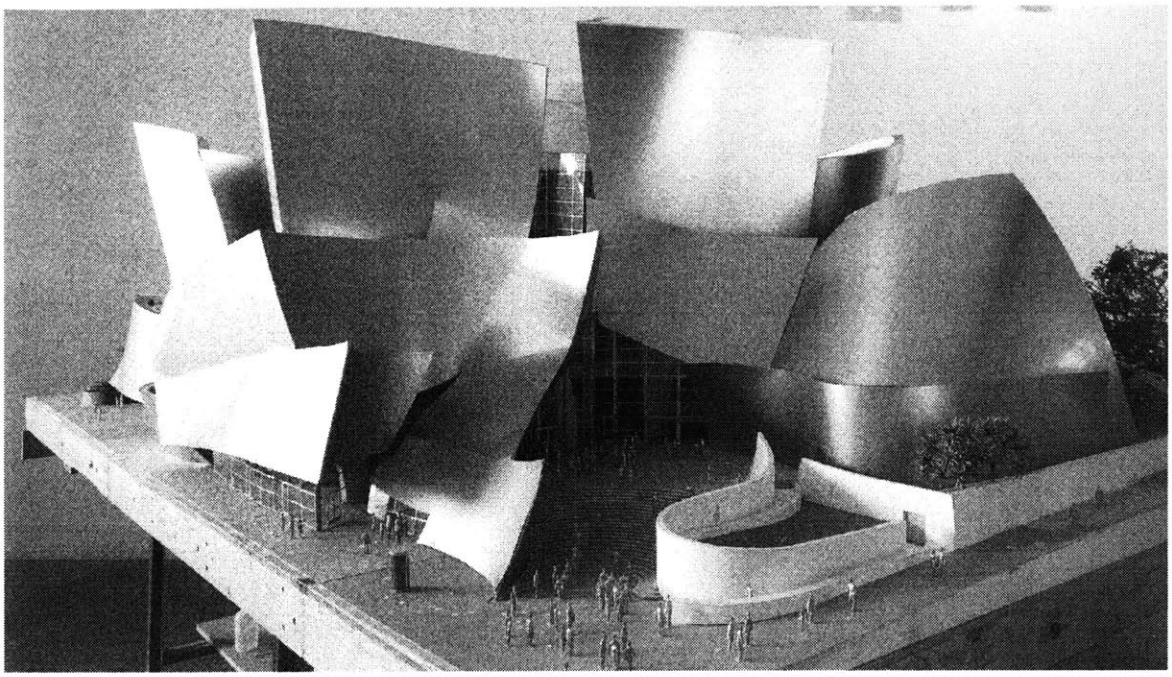


Figure 2-4: Walt Disney Concert Hall. 1989-present. Reproduced from [LM95].

Figures 1-2 and 2-1) – the vast majority of his works have explored design using developable surfaces, surely because they can more readily be produced from various types of sheet metal (see Figures 2-2,2-3 and2-4). Tools to aid in the generation of developable surfaces is an entire research area within Computer Aided Geometric Design field [SF96, Hos98, LP98, PW99, YPM00], and does not fall within the scope of this research.

It is also interesting to note that Frank Gehry's use of computational tools is very similar the that of automotive manufacturers. Based upon my own observations as well as the account provided by LeCuyer [LeC95], Gehry's primary use for his computational tools is related to the manufacturability of many of the components of his architecture. Thus, where many architects use computational tools to aid in design documentation and presentation, Gehry uses them to retain direct control over the manufactured components in his work. In other words, his firm is explicitly producing the geometry that guides the numerically controlled tools that produce various building parts.

Further, due to the complexity of his designs, the computer is a tool that enables him to precisely define the resulting form. Specifically, in my work with him on the Barcelona Olympic Village sculpture, the use of precise computational surfacing tools enabled me to construct a control model of the sculpture which, using high-quality rendering tools, I was able to present to Frank Gehry. The architect was able to feel comfortable that the renderings were, in fact accurate representations of the actual form I had modeled. In this way, he was able to transition from a physical model to a virtual one – this, in turn, allowed him to effectively communicate his design to the firm responsible for engineering the sub-structure of the sculpture.

LeCorbusier was also very interested in freeform surface design [Bes87] and many of

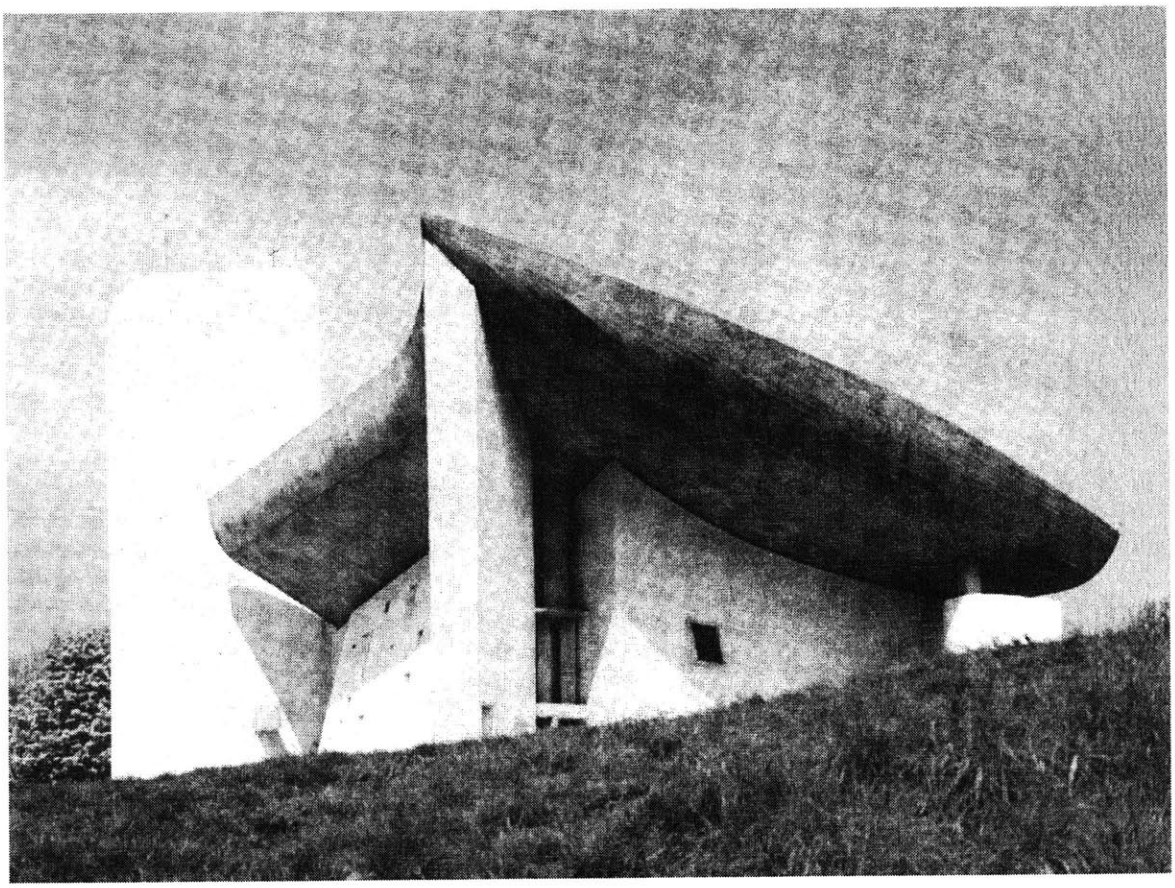


Figure 2-5: Ronchamp Chapel. LeCorbusier, 1950-1953. Reproduced from [Bes87].

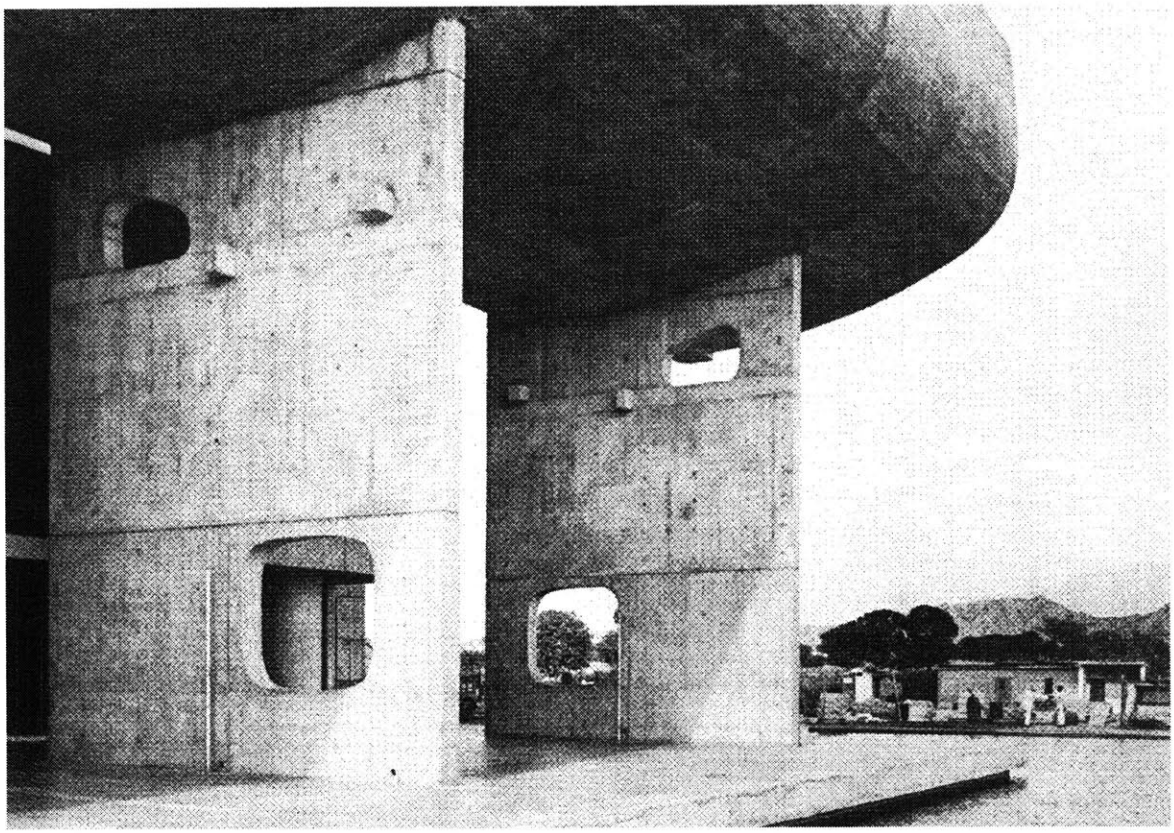


Figure 2-6: Assembly Hall at Chandigarh. LeCorbusier, 1950-1957. Reproduced from [Bes87].

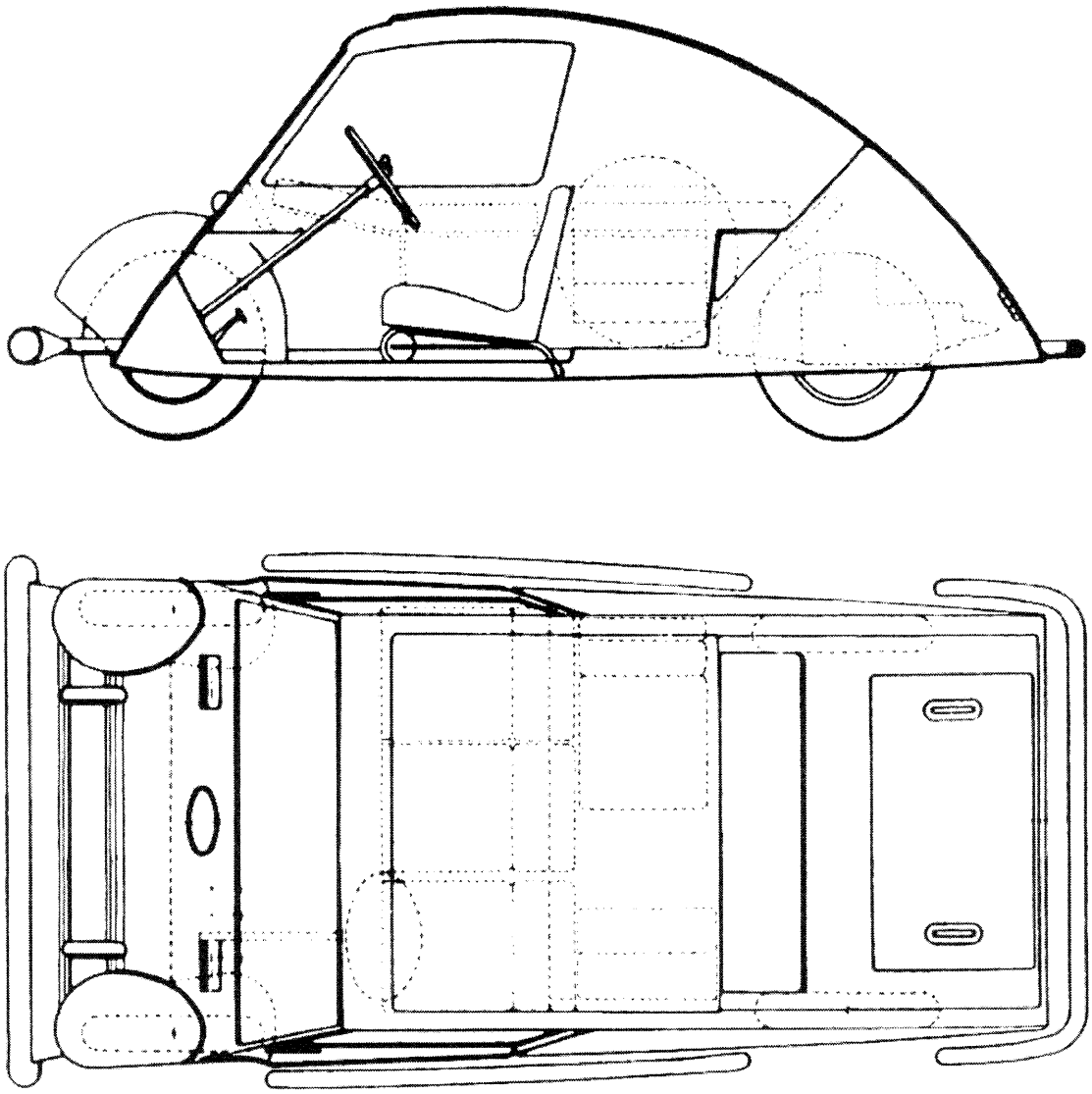


Figure 2-7: LeCorbusier's 1928 design of a car for the masses. His design is significant as it is the first one to forego the running board and separated fenders in favor of leaving more room for the occupants. Reproduced from [Wic87].

his buildings contained freeform surface elements (for example, Figures 2-5 and 2-6). In 1928, LeCorbusier even produced a design for a small car for the masses which involved freeform surface design (Figure 2-7). Also, Günther Behnisch worked with engineer Frei Otto to explore design using minimal surfaces in his 1972 Munich Olympic complex [DD87] (Figure 1-1).

More recently, architects such as Greg Lynn [Lyn99] have been experimenting with computational tools as a medium for their work. While these efforts are still in their infancy, they clearly suggest that architectural designers are on the verge of taking a significant step into the realm of freeform surface modeling.

But it is undoubtedly Antoni Gaudí, more than any other architect, whose work with freeform surfaces reached the greatest maturity [LeC67, de 84, BL99, Zer85]. The range of curved surfaces that manifest themselves is his work is staggering (see Figures 2-8, 2-9, 2-10. He had a mastery of freeform surface sculpture as well as surfaces derived from catenary curves. Figure 2-11 shows a model Gaudí created for the Colonia Güell church.

Again, while it is clear that freeform surface forms have historically been of interest to architects, instances of such shapes are rare. As a result, it is difficult to draw general conclusions about how architects conceive of and develop such forms.

Thus, while this research cannot hope to glean a great amount from architectural freeform surface modeling, it is nevertheless an important area of architectural research as architects will very clearly soon require tools very similar to those required by industrial designers. Further, historically, there is ample evidence that freeform surface modeling is an area of interest to architects in general. Consequently, rather than attempt to formalize architectural design of freeform surfaces it is more fruitful, from the point of view of formulating design metaphors, to examine industrial design.

2.2 Automotive Styling in the Production Process

Automobile styling is a superb example of free-form surface design. As a discipline in which both aesthetically pleasing and functional free-form surfaces are required, it has proven to be an excellent resource for this research. For this reason, the place of styling in the design and development process of a new vehicle must be clarified. In addition, it is important that the actual work done as part of this styling phase be examined carefully. A thorough analysis of this styling work aims to provide insight into the reasons modern computer aided geometric design systems have failed to provide computer tools for styling work. Further, such analysis suggests ways in which computer aided geometric design methodologies and metaphors should be improved in order to redress this failing.

2.2.1 Automobile Body Design and Development

Before delving into the details of the automotive styling process, it is useful to establish its place in the development of a new automobile. Figure 2-12 indicates the typical flow of such a development process. Although the flow arrows suggest a linear evolution, this process is not so organized in practice. Nevertheless, it is approximately true to say that in the overall scheme of things, this sequence does apply – and further, that it applies in most, if not all, automotive development processes.

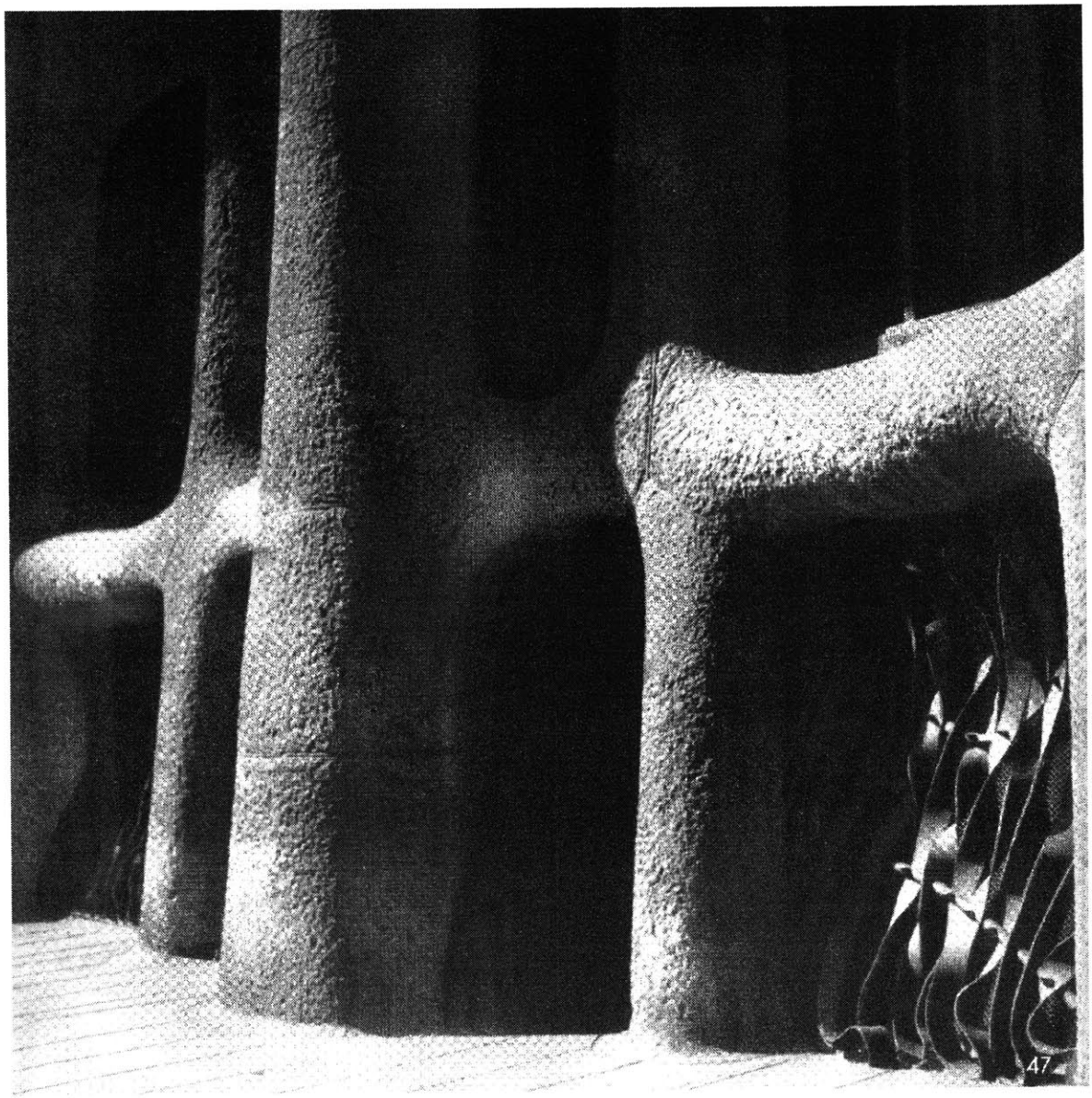


Figure 2-8: Casa Milà facade detail. Reproduced from [LeC67]

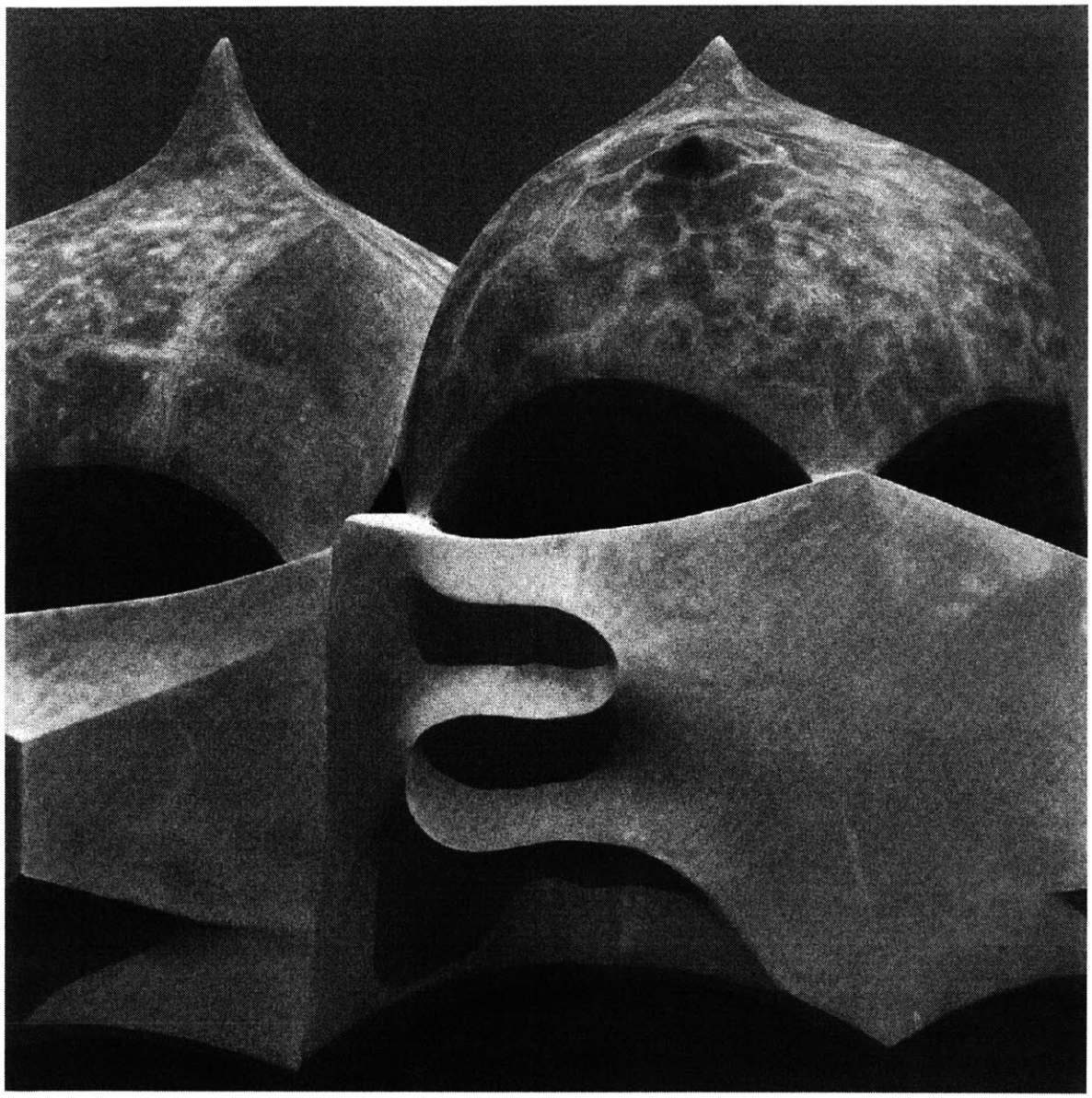


Figure 2-9: Casa Milà chimney detail. Reproduced from [BL99]

Figure 2-10: Casa Batlló facade deatil. Reproduced from [BL99]

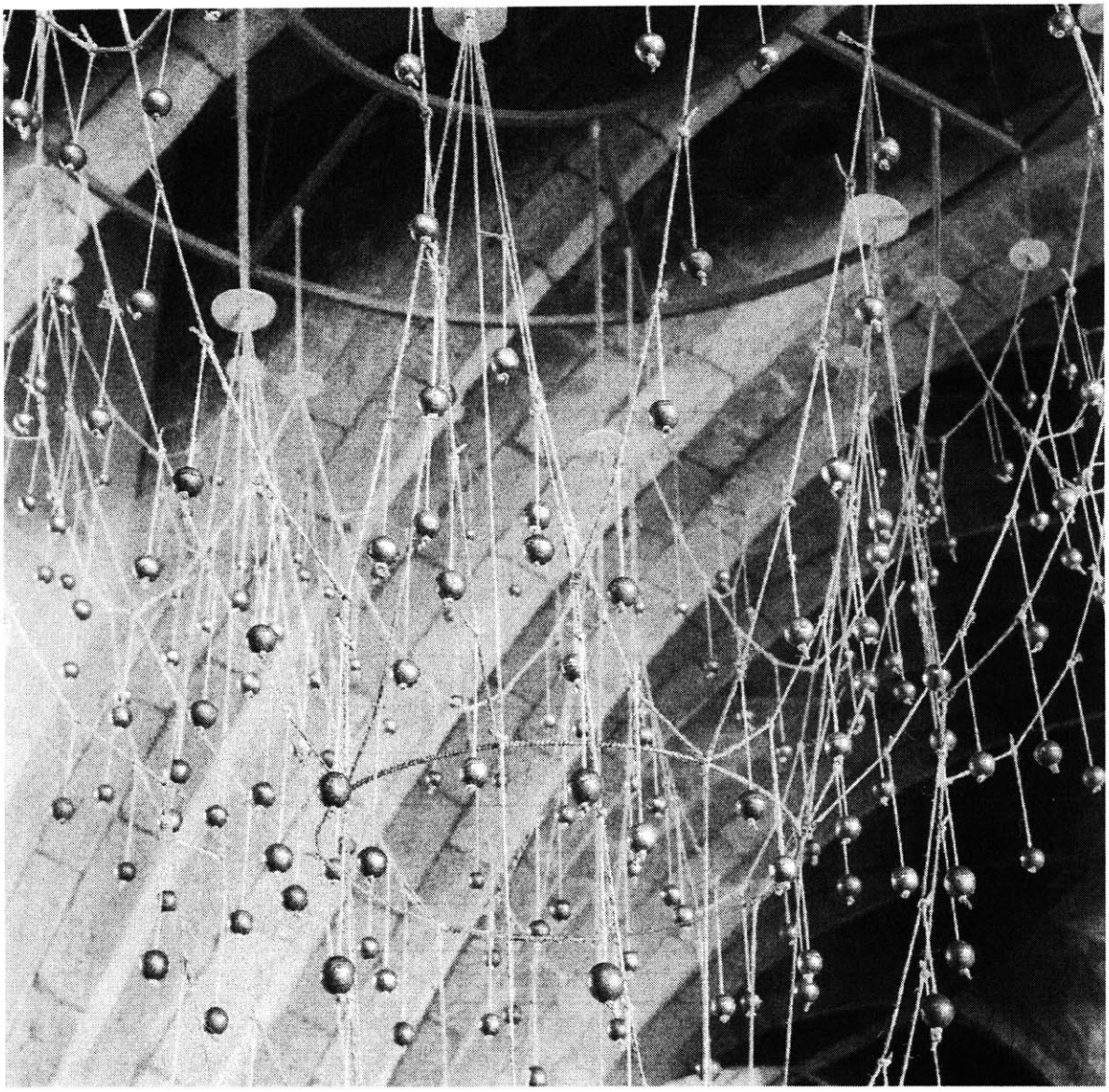


Figure 2-11: Catenary study for the Colonia Güell church. Reproduced from [BL99]

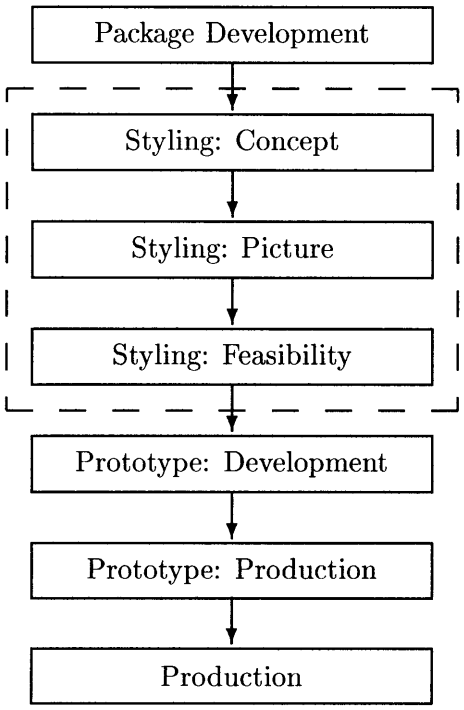


Figure 2-12: Automotive research and development process showing the styling portion surrounded by a dashed line.

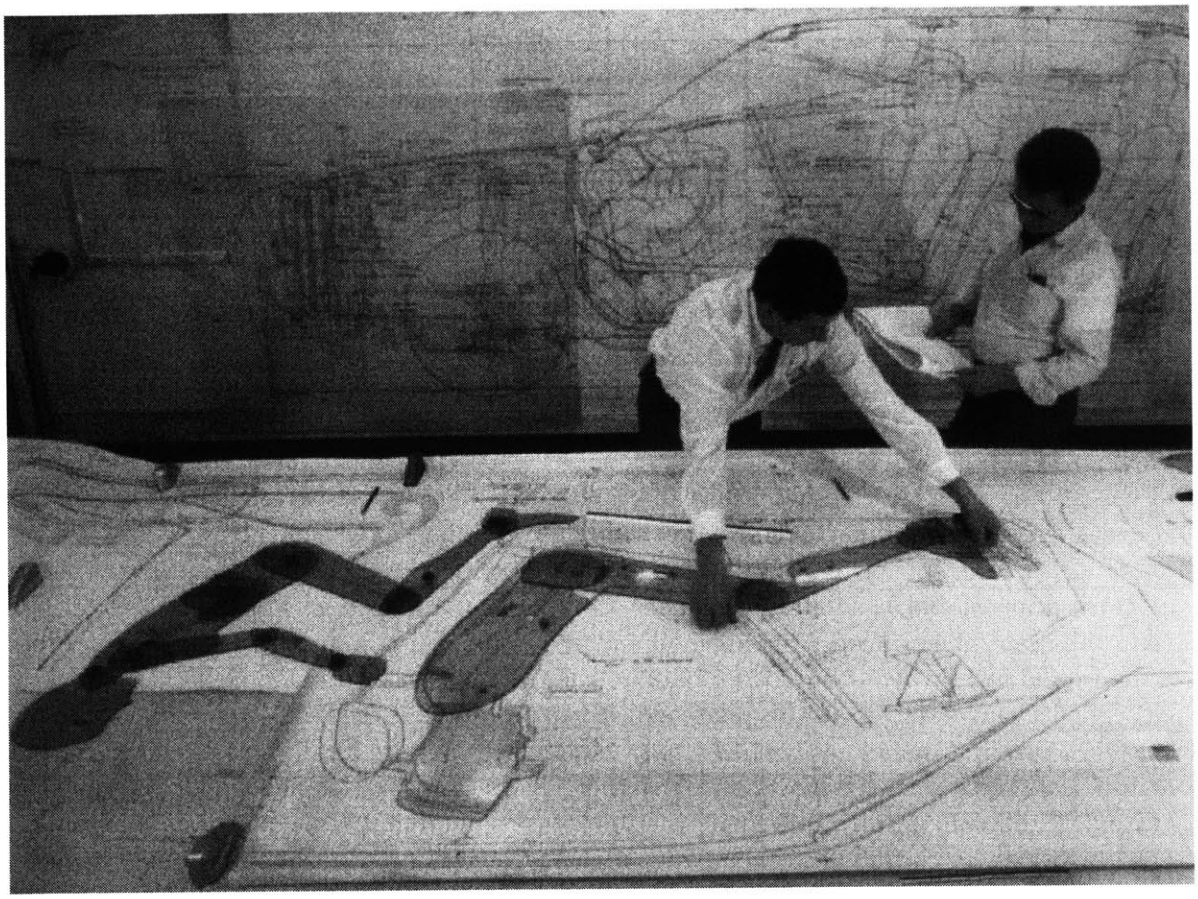


Figure 2-13: Package designers at work. Reproduced from [Wic87].

Of course, an automobile is more than just a body. The drive-train and chassis are also very important components. However, these have longer development cycles than the body does and are thus best thought of as input to the overall body design process. The development cycle for a new drive-train is approximately ten years, while that for a new automobile body is between three and seven years (depending in no small measure on the ultimate quality required in the design).

2.2.2 Package Development

The *package* or *brief* for a new automobile consists of all the planning information available at the commencement of the process. Its function, in a very palpable way, is to set the parameters for all of the subsequent design and development work. Beginning with market analyses, specifications for a new model are developed. These specifications include everything from the overall dimensions of the vehicle, to the features necessary so that the car meets various governmental regulations in all of the various countries in which the vehicle is to be sold. It also includes vital planning information, such as how long each development stage may take so that the final product may be sold profitably.

The output of the package department necessarily consists of large quantities of planning

data as well as actual graphical information. Figure 2-13 shows work on the graphical specifications of the interior of the vehicle. These full-scale CAD plots are the primary input to the styling process. They serve as guidelines within which the designer must work.

Unfortunately, the package for a new model is subject to change even after actual design and development work have begun. While such changes may obviously be critical to the future automobile, they nevertheless make the job of the designer considerably more complicated.

2.2.3 Styling Development

Based on the graphical information provided in the package, the interior and exterior forms of the new automobile body are then designed by the styling department. In the first stage of this styling process, a designer develops a theme or *concept* for the vehicle. She generally conveys this idea to management and others via two-dimensional sketches.

Once the concept has been refined as much as possible, it is then developed into a series of three-dimensional clay *picture* models. The first of these is generally created at less than full-scale. The purpose of this model is to develop a first approximation of the three-dimensional form which best corresponds to the concept earlier developed. Designers work closely with clay modelers in this phase in order to best express the form three-dimensionally.

Once this first model has been created, a subsequent picture (or design) model is created at full-scale so that the proportions are clearer. The reason for this next model is that it is extremely difficult to fully appreciate the proportions of an automobile from a scale model. Once a design is clarified and complete, a three-dimensional clay or fiberglass *feasibility* model is created. This model becomes the control information for subsequent interior and exterior surface development.

Although the three-dimensional feasibility model is the control information, much of the subsequent development requires computational processing of the surfaces. Thus, styling departments often provide three-dimensional sampling services so that they can forward computer data to subsequent departments rather than merely a physical model.

2.2.4 Prototype Development

In this phase of the development of an automobile, the design is converted to actual body panels with structural and aerodynamic properties. Initially, the surfaces generated in the styling process are converted to computer aided geometric design data. Reconstructing surfaces from such data has been a thriving area of research within the computer aided geometric design field.

These exterior body panels are known as *Class I* panels and they are subsequently *offset* (another important area of CAGD research) in order to assist in the definition of the *Class II* panels – the unseen panels which help to give the automobile structural integrity. During this phase, computer simulations are run in order to test aerodynamic, structural and crash properties of the design. These simulations are generally run on finite element meshes which discretely approximate the surfaces which make up the design. Once completed, full scale prototypes are constructed which are made of the same materials as the final design. These have many purposes, not least of which is to be tested in crash simulations.

2.2.5 Final Prototype of Vehicle and Production

Once as many errors as possible have been removed from the design, a final prototype is built. This serves as the control model for production purposes. This prototype includes a great deal of production information including such details as spot-welding locations. When this final prototype of the automobile is complete, it remains to prototype the actual manufacturing process by which the body is to be produced.

2.2.6 Production

Finally, the production phase is reached. The plant is retooled according to the specifications developed as part of the production prototype development, and then production begins. Of course, it is never the case that there are no further problems to be resolved, but it is easy to understand the immense expense of changing designs at this late point.

2.3 Automotive Styling

With a rough overview of the preproduction work on a new automobile in place, it is now possible to contemplate the styling process more carefully. The purpose of the investigation is not merely to master the basic sequence of events involved in the styling design of a new car, but much more importantly, to reach some conclusions concerning the types of tools and thought-processes involved in this highly subjective process.

Thus, it is crucial to this research that as much as possible in known about precisely how designers and modelers approach the task of designing aesthetically pleasing, as well as functionally correct, exterior surfaces for automobiles.

Parallel to this exterior surface work, designers also develop the interior of the vehicle. The styling process for this is very similar to that of the exterior in terms of the process flow. Indeed, most of the tools used in interior and exterior styling are the same.

In the interests of simplicity, this account will concern itself with only the exterior surfaces.

2.3.1 Concept Development

Before addressing the details of designing the automobile, designers attempt to develop a small set of *concepts* with which they intend to imbue the vehicle. Thus, they first attempt to suggest – not scientifically identify – the character that the car is to have. Of necessity, many crucial design issues are passed over at this early styling stage in favor of developing these themes for the design. Many of these themes can be approximated in language with adjectives such as fast, aggressive, elegant or even playful. Figure 1-4 shows an example of an early concept sketch. Note that the sketch depicts more of a caricature than an automobile. The designer is able to take a great deal of liberty with the two-dimensional images she creates. However, the designer is ultimately responsible for evoking a feeling in the viewer which she can, in fact, deliver in three-dimensions as well. This dynamic produces an interesting tension in this work between idea and reality.

Along with these evocative representations, the designer also produces a number of high-quality renderings of the possible final outcomes (see Figure 2-14). Although these more

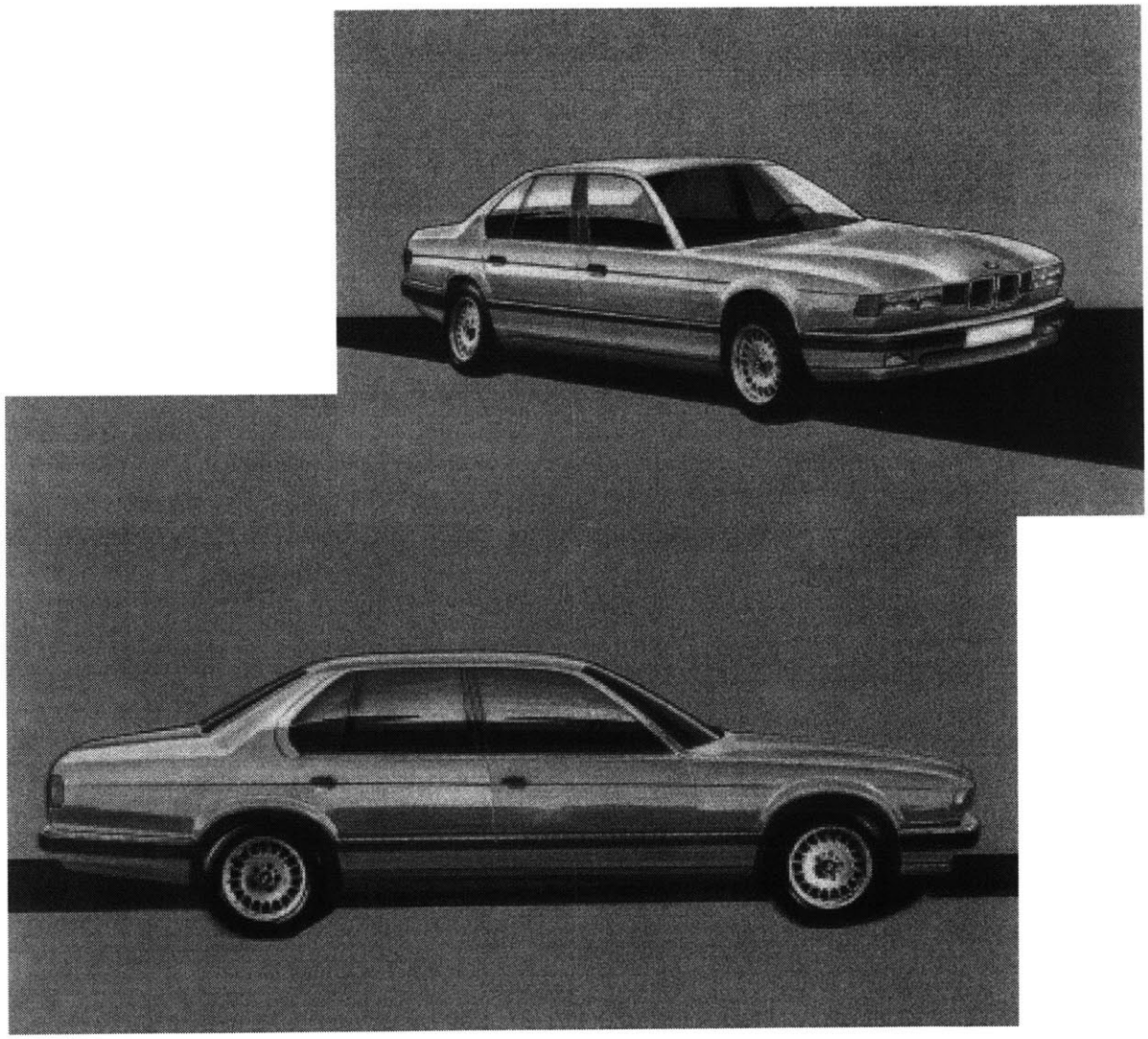


Figure 2-14: BMW 7-Series. Early concept sketches. Reproduced from [Wic87].

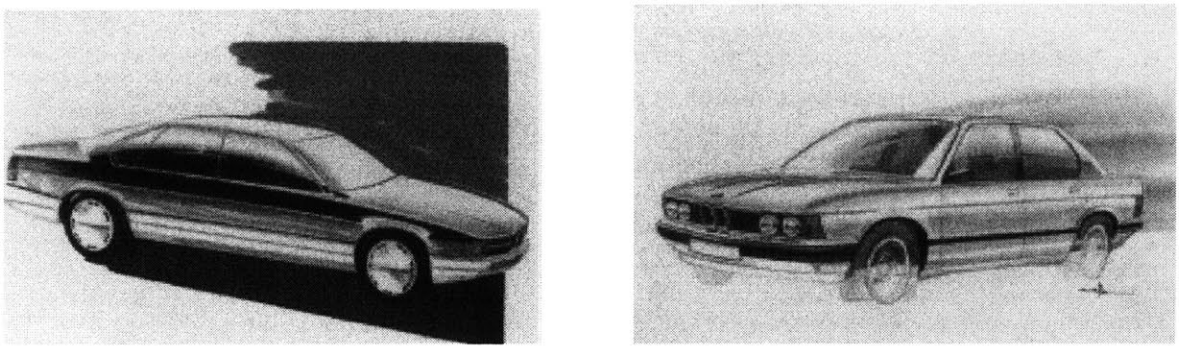


Figure 2-15: Two early BMW 7-Series concept drawings. The left sketch suggests sleekness while that on the right evokes feelings of power and speed. Reproduced from [Wic87].

refined sketches are more realistic, they are nevertheless conceptual designs, and, as such, they only pretend to depict photo-realism. Thus, these renderings are designed to present the design in the best possible light – even if this light can not exist in three-dimensions.

The primary medium in this early phase is paper. These early two-dimensional sketches may be created with pencil, pen, markers, pastels, watercolors or any other tool of the designer's choosing. In short, the image conveyed is much more important than the means of conveyance at this early stage. Thus, some designers even work on systems such as Quantel's PaintBox which allows them to create two-dimensional digital images at the resolution of high-definition television (HDTV) or greater.

Sources of inspiration vary widely from designer to designer, as well as from project to project, and from firm to firm. Beyond preferences of the designer, influences on any given design may include previous models of the same name, other cars, or occasionally developments outside of the automotive field.

Regardless of the influences, designers must first develop a character for the car. In a new version of an existing model, this character is usually largely predetermined. In a new model, however, the designer generally has more latitude. In addition, companies have certain identities which must be preserved (many pretentiously refer to this as *Brand DNA*). For example, the radiator grille of a BMW has a distinctive *Nire* (German: liver) form. There are other more subtle details as well. Clearly, the designer must not only be aware of these but must also understand their evolution in the history of the marque. Other automobile manufacturers such as Ford, for example, place much less emphasis on consistency across their entire product line – or along their historic time-line, for that matter. Certainly, BMW produces cars for a much smaller cross-section of any given population than does Ford, and, not surprisingly, this influences the extent to which such consistency is possible or even desirable.

Synthesizing these various inputs to the design process with a new idea for a car is truly the job of the automotive stylist. Beyond that, it is critical that she be able to convey this concept graphically and that it be in the appropriate Zeitgeist (see Figure 2-15). It is not easy for the designer to anticipate what will be in fashion seven years into the future and yet this is precisely what she must do. It is likely that, to a great extent, their work defines fashion as much as it follows it. Thus, their work may not be so much to anticipate fashion,

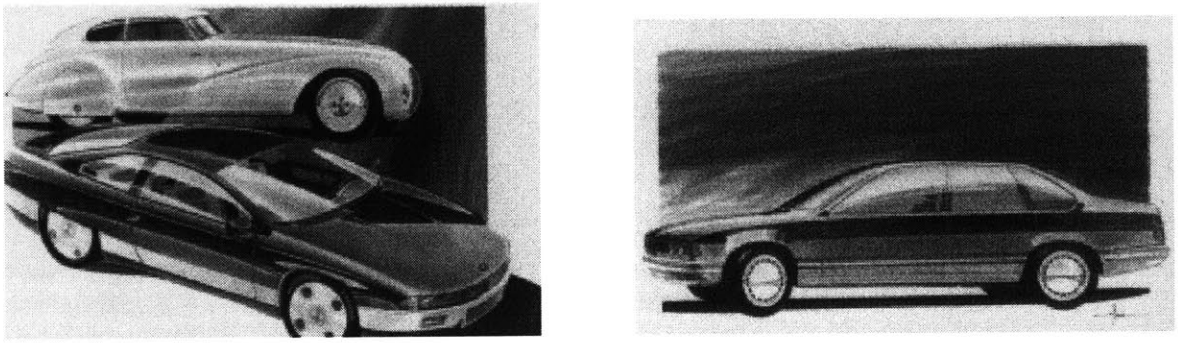


Figure 2-16: BMW 7-Series. The left sketch explicitly provides the influence while that on the right attempts to capture the slender elegance of the BMW 7-Series. Reproduced from [Wic87].

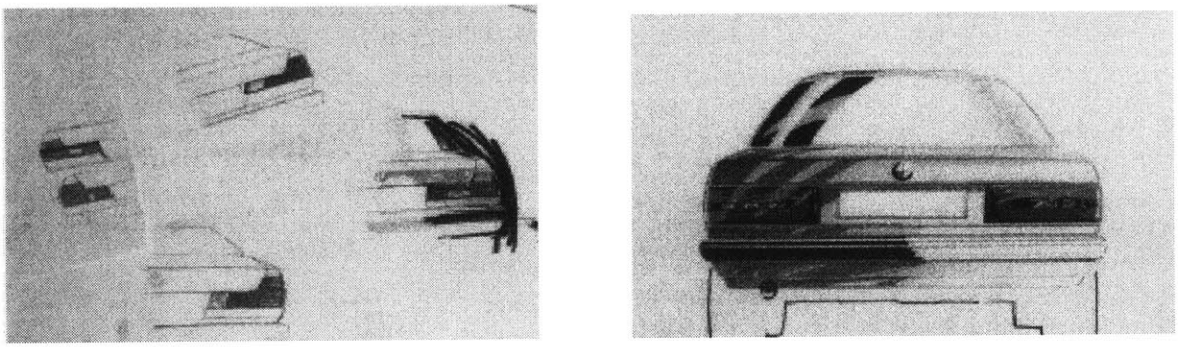


Figure 2-17: BMW 7-Series. Tail light studies. Reproduced from [Wic87].

but to create it.

Perhaps it is best to describe this concept-development work by means of some examples of two-dimensional images created as part of this early phase. Figure 2-16 shows two sketches created to demonstrate that the stylist intends to develop a design which follows a theme developed over time. Figure 2-17 shows a number of studies to develop tail light schemes for a new design.

It should be clear that this early conceptualization has much in common with impressionism and expressionism. That is, it is really about evoking emotions which will place the automobile in a favorable light. Specifically, the format of these sketches is very often a front or rear 3/4 perspective view or a face-on view of the side, front, or rear of the vehicle. Perspective is used extensively to exaggerate the feeling conveyed in the sketches (see Figure 1-4). Reflections lines are also used to convey a sense of the shape of the surfaces. Along with reflections, surface highlights are also extremely important to this task. A surface highlight is technically also a reflection although it is of a special type. Highlights are the result of a sharp "bend" in the surface. More precisely, this means that widely different surface normal directions are very near to one another along the surface. Thus, the surface seems to "catch" a great deal of light along this crease in the surface – hence we see a bright

curve along such a crease.

An extremely important observation to be made at this point is that these sketches are by no means accurate in any sense. Their sole purpose is to convey feelings; thus, to move from these feelings to more fully defined three-dimensional data is no mean feat. In fact, this is the task in the next phase of development. Based primarily on these sketches, a three-dimensional clay model is created – generally at less than full scale.

The creation of a coherent concept for a new automobile is vital to the overall design as well as the way the vehicle will ultimately be perceived on the road.

2.3.2 Picture Model Development

Once the management has chosen a number of schemes for continuation beyond the conceptual development phase, the next step in the process is to give the idea three-dimensional form. In this endeavor, each designer works with a clay modeling team. These modelers are professional sculptors and the nature of their training is extremely applied. Often, companies prefer to provide this training in-house because each company employs modelers in slightly different ways. The first job of the designer is to interpret the sketches used in the previous phase and produce what are known as *tape drawings* (often just *tapes*), or *key-line drawings.*

In the creation of the tape drawings, the designer must commit to a shape for the automobile. Up to this point, there has been considerable leeway given to the designer in the name of the creation of an idea. Now, it becomes the job of the designer, together with the modeler, to convert the ideas developed thus far into unambiguous surfaces with real physical properties. The first step is to create these tape drawings because modelers can use these to create a *proportion* or *blocking* model rather quickly. It is important to note that while the designer must commit to these tape dimensions at this point, they will almost certainly change once the first model is built.

The tape drawings are always at full-scale and usually they are created as overlays on top of the package plots (see Figure 2-18) which are provided by the package development team. They are always posted on a wall very near to where the model is being sculpted. These tapes correspond to the two primary sections and a plan of the car, but they are developed to include the three elevations. Thus, the designer creates these tapes using the package information as a reference.

The medium for the creation of a tape drawing is, not too surprisingly, tape on mylar. It is a special tape which has a fabric surface on one side and glue on the other. It is available in different widths in order to simulate different bending stiffnesses. Naturally, the thicker the tape the more resistant it is to bending. In this way, designers are constrained to designing curves which are "nice" in some way. By choosing different tape thicknesses, they are able to control curvature changes in a very intuitive way.

The information in the package and on the tapes is used to create a metal frame for the model (see Figure 2-19). This frame is typically created in the metal shop. It should be noted, however, that a model frame need not be made of metal if the scale is sufficiently small. Next, a rigid porous foam material (polystyrene) is applied to the steel frame to a depth of no more than three to five centimeters (at full scale) from the eventual surface (see Figure 2-20). By using this method, the clay can be packed onto the foam to make up the

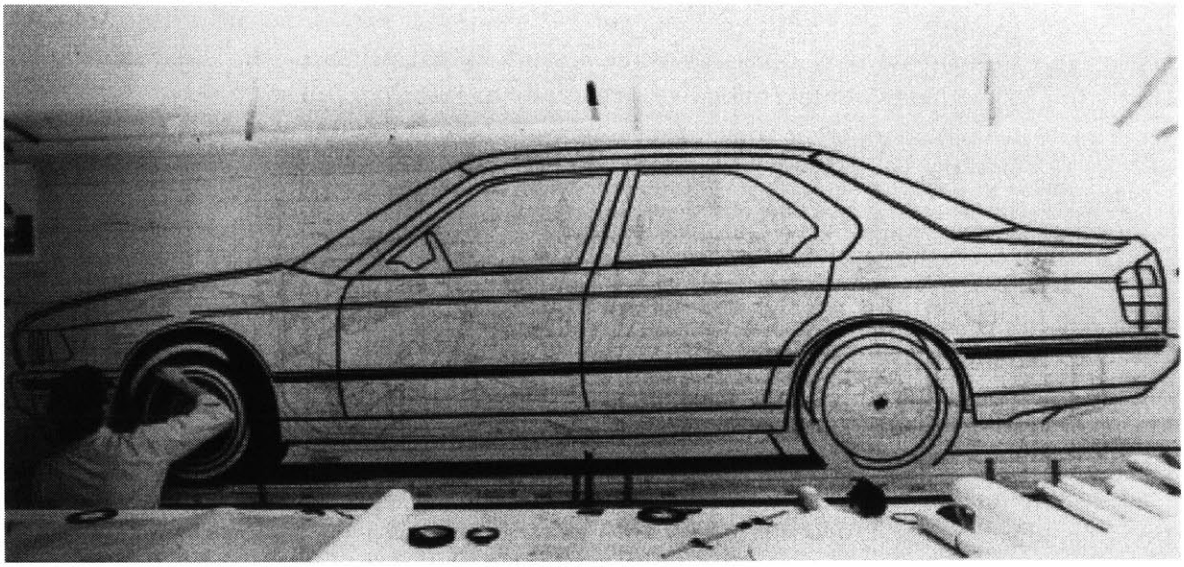


Figure 2-18: Designer working on a tape. Reproduced from [Wic87].

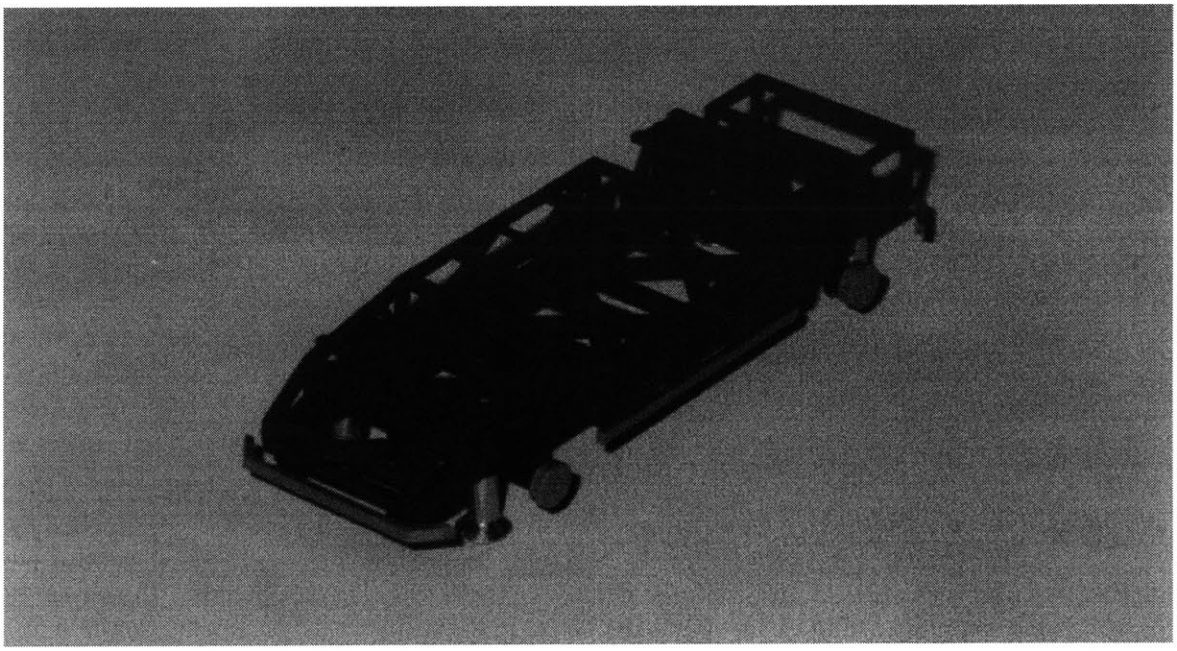


Figure 2-19: Steel frame for the clay model.

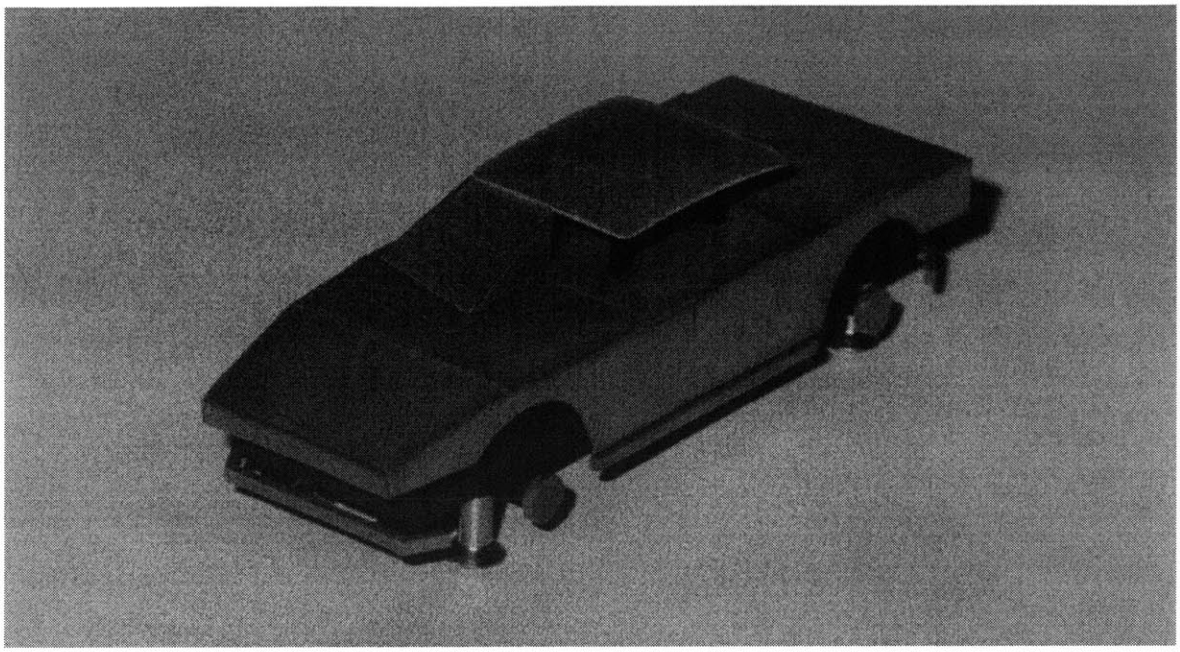


Figure 2-20: Application of rigid foam material to frame.

difference. In subsequent models, the clay thickness is reduced because any changes in the surfaces at later stages will be of smaller magnitude.

Thus, once the modelers have packed on enough clay over the entire model, the actual clay modeling process begins. Figure 2-21 shows a model depicting two stages of the process. On one side, the clay has not yet been applied, while on the other, significant work has already been done to refine the surfaces. On the side to which the clay has yet to be applied, inverted dimples in foam core are visible – these prevent the clay from breaking away from the near-vertical surface. Once the surfaces have been refined to the point at which the designer is satisfied, detail is added to the model (see Figure 2-22). Thus, the designer is also responsible for creation of the exterior detail of the car.

Not surprisingly, much of the work in creating this first three-dimensional model is in the refining of the clay surfaces. This clay modeling work is rich in suggestive ideas for computer aided free-form surface design tools. A number of these are developed in this research.

Clay Modeling

Although they are ultimately subject to many changes, the tape drawings are not modified significantly while the first approximate model is created. This is because it is far more important that **any** model be present than that all the details be resolved from the start. Once a model has been created, the more interesting and meaningful work can begin.

In as much as they are accurate, tape drawings tell the modelers quite a bit about the overall form of the automobile. When the tape drawings do not contain enough information,

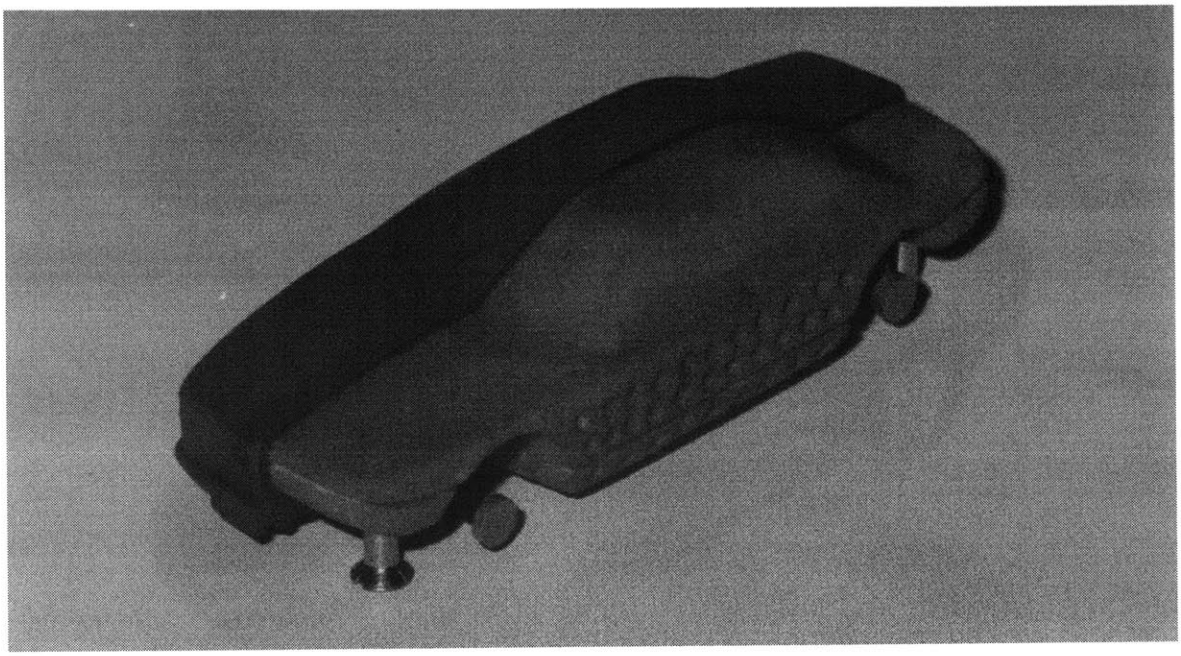


Figure 2-21: Application of clay and the development of surfaces.

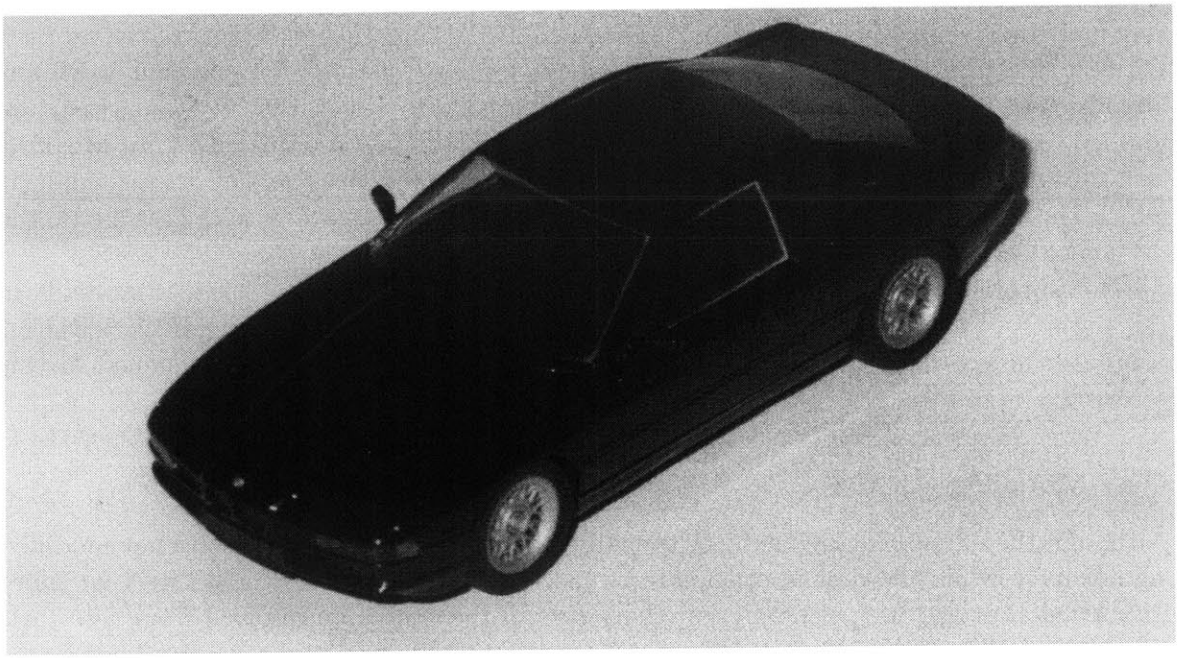


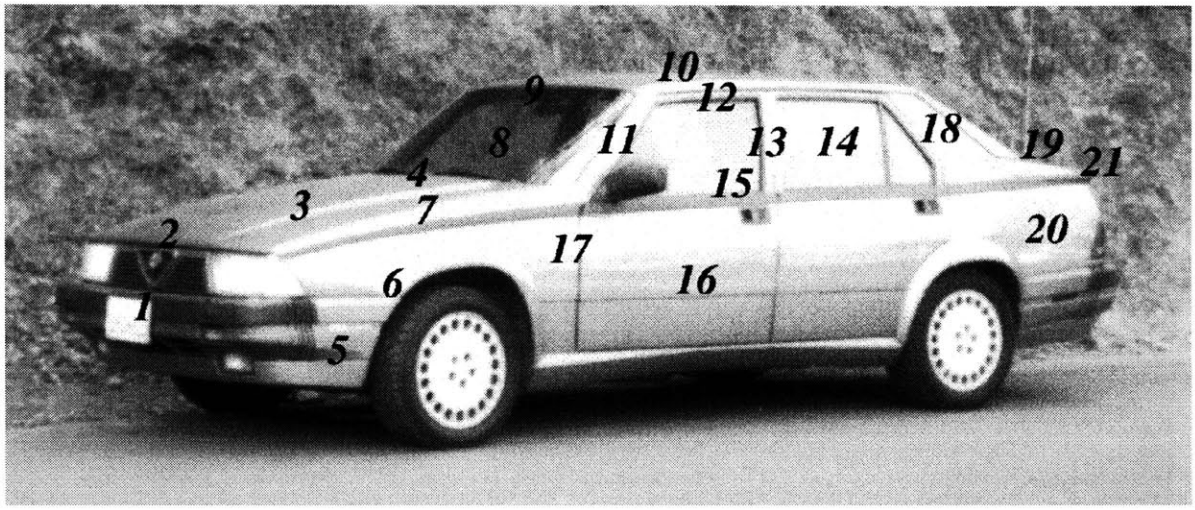
Figure 2-22: Finished model of exterior.

Figure 2-23: View of a BMW Modeling Studio. Reproduced from [Wic87].

modelers will discuss the problems with the designer as the work proceeds so that the model meets the requirements of the designer – who is, after all, ultimately responsible for the design.

Setup Before discussing the actual process of transforming two-dimensional ideas to three-dimensional forms, it is useful to have a sense of the layout in the clay modeling studio. As the models and tapes are often large, space is important in the process.

The modeling studio space is generally laid out so that the tape drawings are hanging near the model as is suggested in Figure 2-23. It is important in this process that the tapes drawings and the model being created can be viewed from a reasonable distance. On either side of the model, three-dimensional layout machines can slide along the length of the model. These are used to create a reference point in space. This reference point will be used to recalibrate the layout machines as their accuracy diminishes linearly in the amount that they are moved. Thus, they are generally recalibrated between each measuring task. These layout machines are used extensively during the modeling process as the general lack



1. Front End
2. Front End of Hood
3. Hood
4. Rear End of Hood
5. Bumper Side
6. Front Fender
7. Crown Line
8. Windshield
9. Upper End of Windshield
10. Roof
11. A-pillar
12. Drip Line
13. B-pillar
14. Side Window
15. Belt Line
16. Body Side
17. Accent Line
18. C-pillar
19. Trunk
20. Rear Fender
21. Rear End

The term Accent Line is often Character Line or Design Line and can occur anywhere.

Figure 2-24: Automobile anatomy.

of precise information available to the modelers makes it imperative that they make use of what little precise information they have.

Finally, in order to better understand the automotive clay modeling process, it is helpful to be aware of the terminology associated with the anatomy of a car. Figure 2-24 gives an overview of this terminology. These terms are not unique and this list is not complete: However, the terms in it are used rather consistently and provide a good point of reference. In the discussion that follows, these terms will be used whenever possible.

Tools. Along with the terminology for describing the automobile itself, it is important to be familiar with the terminology associated with this industrial sculpting.

The clay itself is of industrial grade with very little grit, and has the critical property that

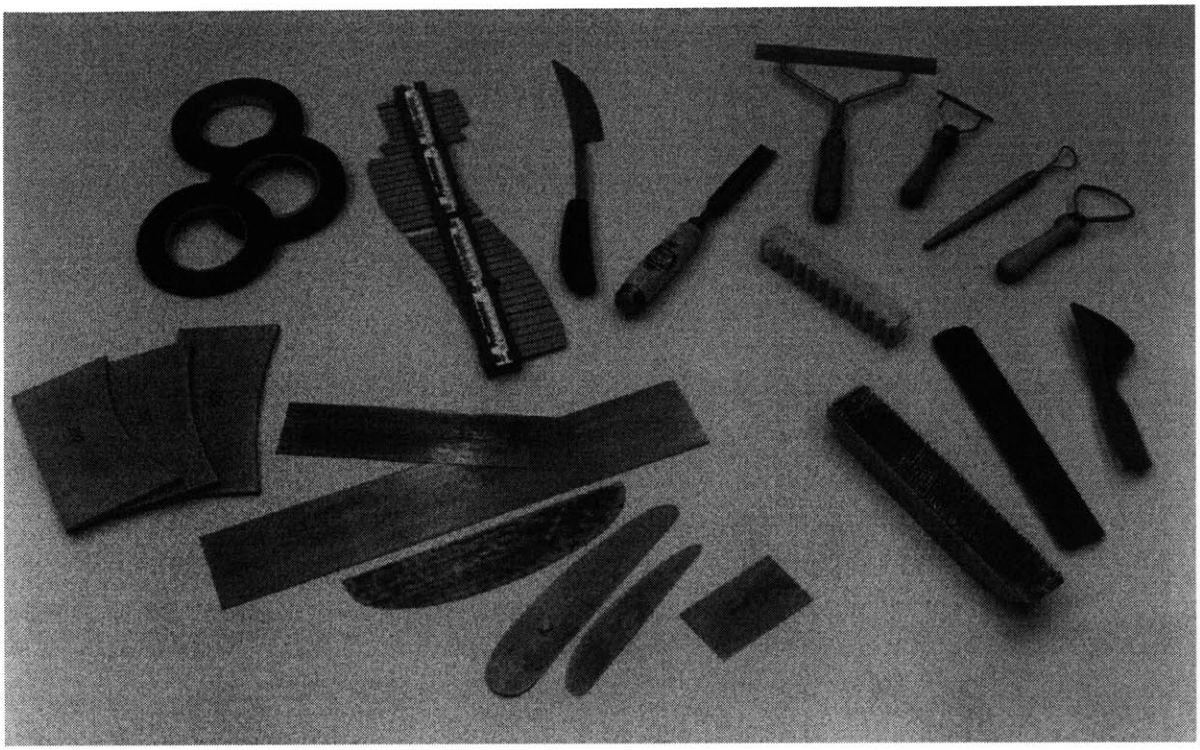


Figure 2-25: Master Modeler's Tools: A Selection

it is quite soft at temperatures between 45^oC and 60^oC, while it becomes reasonably rigid at room-temperature. Thus, portions can be applied after they have been heated up to the appropriate temperature and they can be worked until they approach room-temperature. Also, a heater (similar to a hair-dryer) can be, and often is, used to heat a surface so that it may be modified after the clay has cooled.

The tools used by modelers are basically the same as a sculptor would use to create clay sculpture. Of course, automobile modeling requires more accuracy than does normal sculpting, but the tools used for the actual manipulation of the clay are much the same. Figure 2-25 shows a small selection of the tools used by these industrial modelers. These tools can roughly be broken down into five different classes: *rakes*, *files*, *blades*, *templates* and *true-sweeps*. The first two are primarily used to create surfaces while the last three can be used to both create and inspect surfaces. Each of these tool classes will be described in turn.

Clay is applied by hand to the model where a surface is to be created or modified. Taken from special ovens, fist-sized pieces of clay are rubbed onto the existing surface. A great deal of pressure is required in applying the clay as it is very important that it be packed on such a way that there are no air pockets trapped beneath the surface. Such air bubbles would compromise the integrity of the eventual surface and it would therefore need reshaping.

Once clay has been applied to roughly the correct level – slightly higher than the ultimate

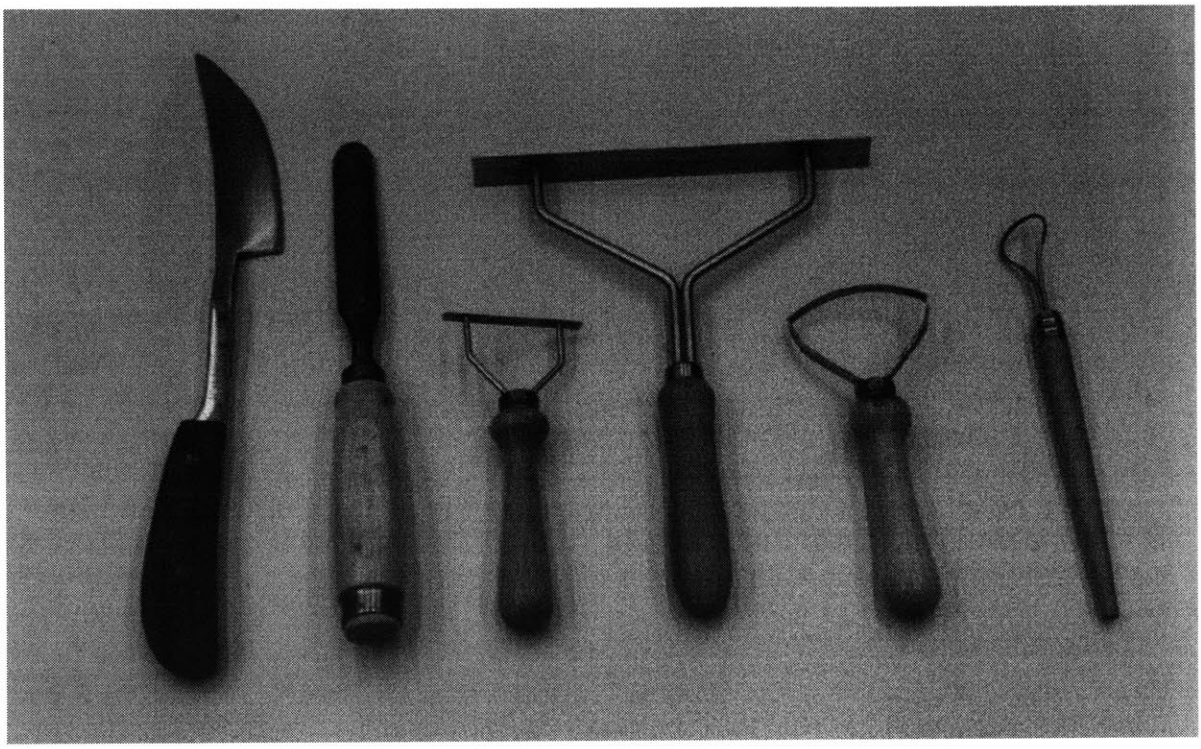


Figure 2-26: Rakes: A Selection

surface – *rakes* – some examples of which are shown in Figure 2-26 – are used to remove clay until the surface is approximately at the correct level. In this case, the correct level is still somewhat above the level of the target surface. Thus, with a rake, the modeler can remove large irregularities from the clay surface. After the raking is complete, the surface is by no means smooth, but there are no large bumps in it.

Next, *files* (see Figure 2-27) are used to smooth the surface even more. The contact-surface of a file is very similar to the fine side of a cheese-grater. After the use of the rakes and files, the surface should approximate the intended surface very closely, so that the only remaining task is smoothing the surface.

This final smoothing is accomplished with *blades*. Only spring steel blades are shown (see Figure 2-28), but modelers often also use wooden ruler-like sticks to smooth surfaces. Whether of steel or of wood, these tools are all used in the same way. Modelers exploit the spline nature of these blades so that, by using them, they are able to create surfaces with nice *accelerations*. The acceleration of a surface is the term designers and modelers use to refer to the C^2 properties of the surface. Obviously, though, this continuity is only enforced to the extent that the modeler uses the blade to check the acceleration in this way. In fact, the appearance of the surface to the eyes of the modeler and designer is much more important than its mathematical properties. Nevertheless, these two notions often overlap.

When a section of a surface remains unchanged in one direction, *templates* are often used to create those regions. Some examples of templates are shown in Figure 2-29. Custom

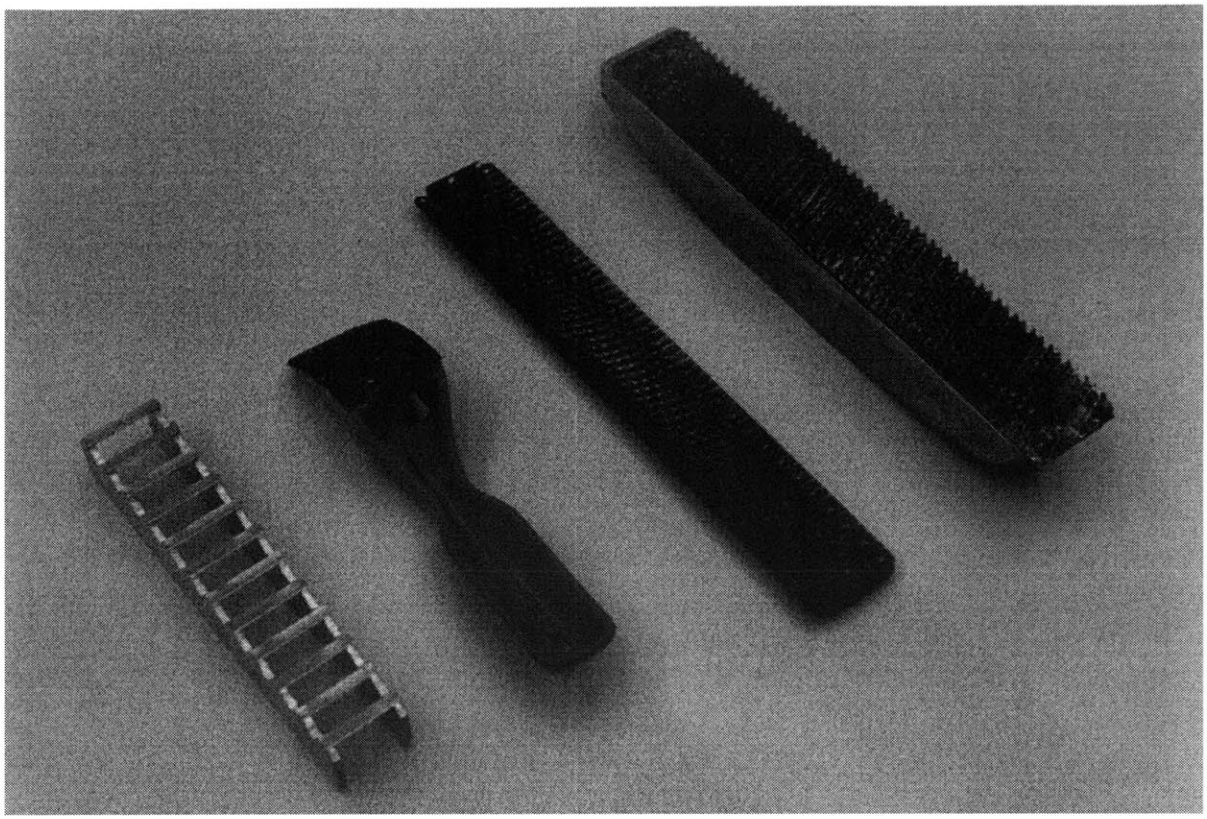


Figure 2-27: Files: A Selection

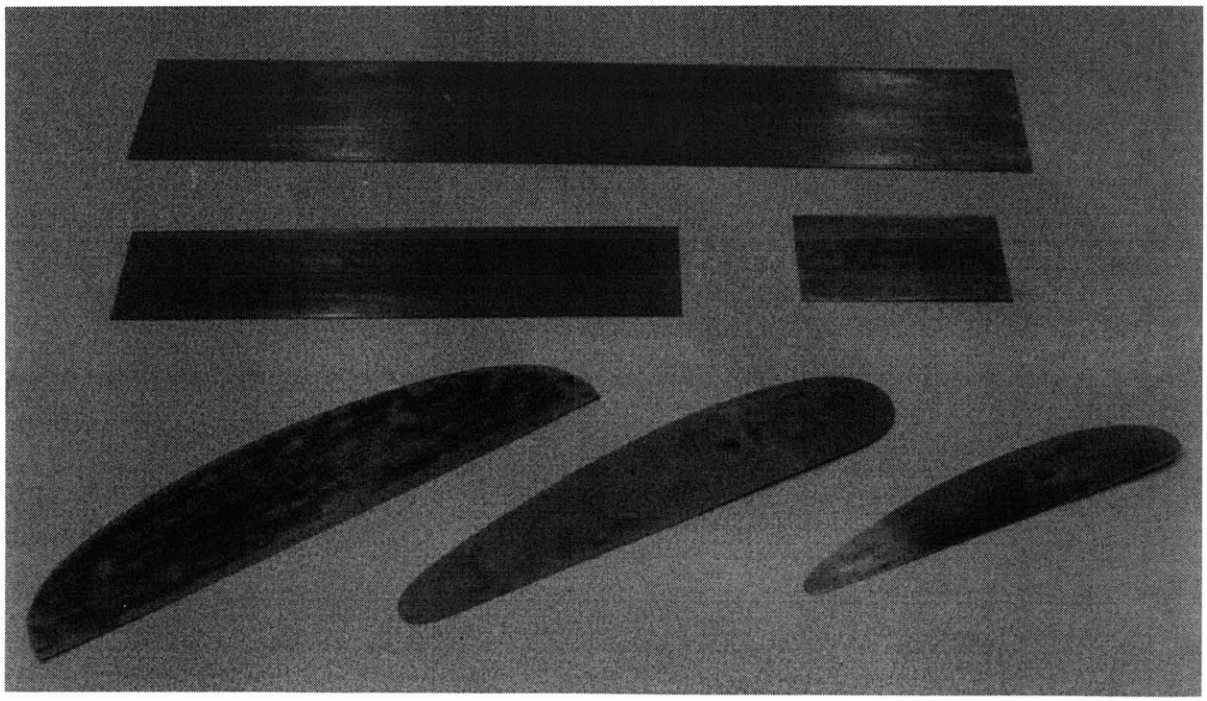


Figure 2-28: Blades: A Selection

Figure 2-29: Templates and curves.

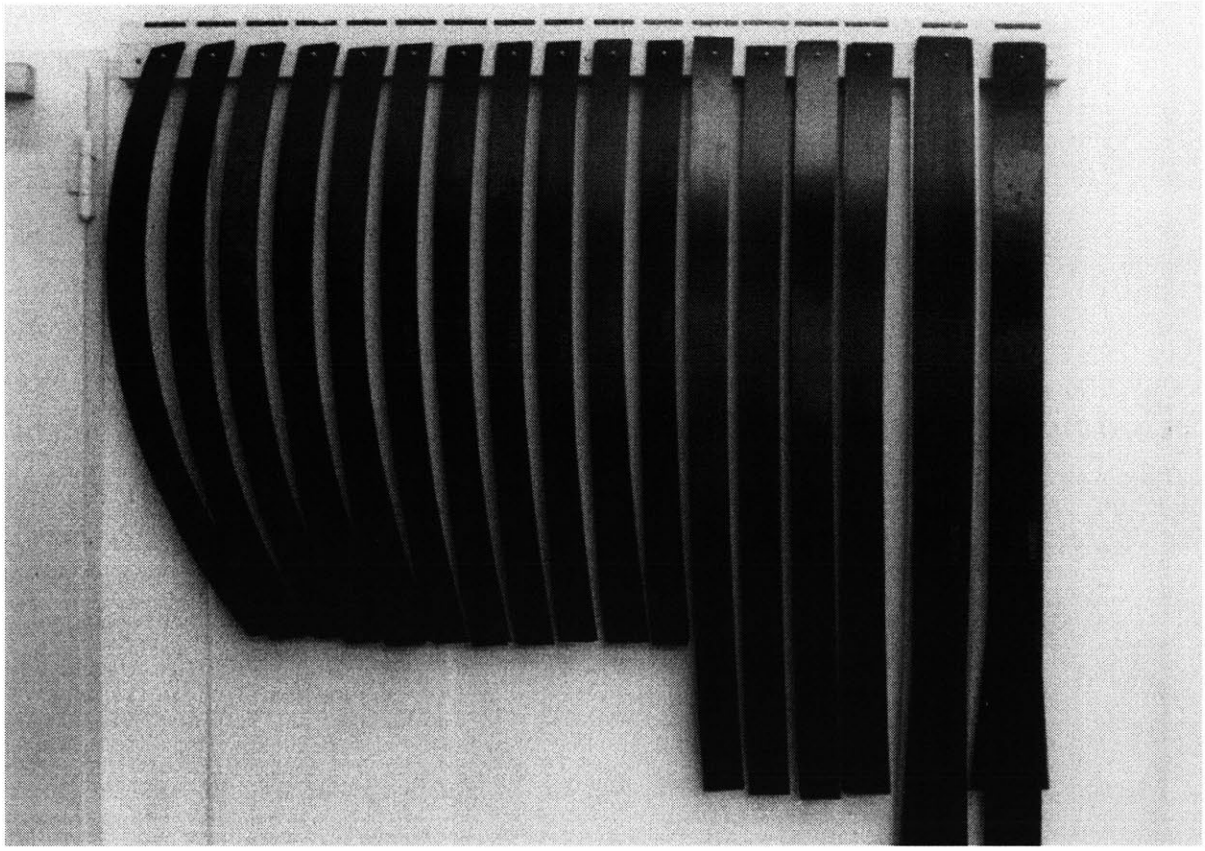


Figure 2-30: True-sweeps

templates are actually built for important large regions such as the front and rear ends of the car. These curves as well as the other tool-types depicted in the figure can also be used to interrogate a surface to ensure that it matches the tapes. Similar uses are found for true-sweeps (see Figure 2-30). These are occasionally used as templates but are far more frequently used as measuring instruments.

Here, it is worth noting that non-trivial differences exist between how these tools are used from company to company. Based on the account of clay modeling at the Nissan Design Center provided by Yamada [Yam93] and observations made at BMW during the summer of 1995, Nissan uses true-sweeps far more extensively in the creation of surfaces than does BMW. This difference has significant implications for the types of surfaces which can be created. As true sweeps have a fixed radius, surfaces created with them tend to have a static appearance. On the other hand, surfaces created with blades can be made to have a more dynamic and tensioned appearance. In other words, the designer is able to modify surface accelerations more readily at BMW than at Nissan.

Of course, the modeler may choose to use any other tools which help in the creation of a surface. For example, fishing line is often used to strengthen a sharp edge. Similarly, when templates are used to generate a swept region, thin metal spring strips are often embedded in the surface to act as guide rails in order to protect already correct regions of the surface.

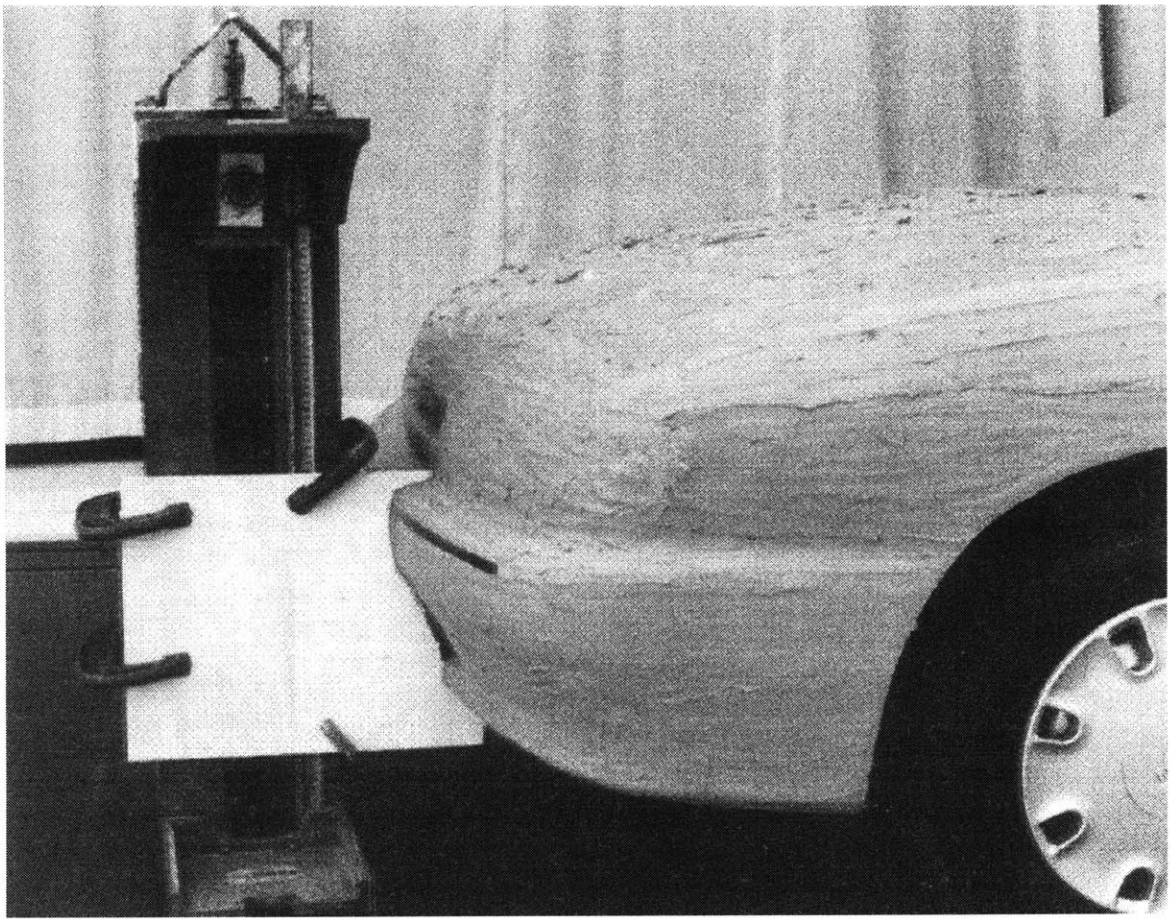


Figure 2-31: Template used to sweep the front end. Reproduced from [Yam93].

From Tapes to Three-dimensions: A First Attempt. With both the contents of the modeler's tool-box described and the setup of the modeling studio defined, it is now possible to describe the process of creating a clay model.

As suggested above, after the frame and foam have been constructed, clay is hand-packed onto the foam over the entire car body. Next, design of the surfaces for which precise information is available in the tape drawings begins. Foremost among these surfaces are the roof, hood, trunk, front end, rear end and body side (refer to Figure 2-24).

The front and rear ends are sculpted using custom-made templates attached with clamps to a precision bridge as shown in Figure 2-31. These templates are traced from the tape drawings onto rigid boards. They are subsequently cut so that a negative of the form remains. Similarly, the path curve is traced from the tape plan. These templates are positioned in relation to the actual model using the car's coordinate system. Next, a sufficient amount of clay is packed onto the model so that dragging the template along the path curve will strip away excess clay to leave the desired swept surface behind. A great deal of care is taken in constructing and using these templates, as they are part of the

small amount of precise information available to the modeler at this early stage of model construction.

In parallel, the roof, hood and trunk surfaces are created. These are somewhat more complicated to create than the template surfaces as they are significantly less constrained by the tapes. However, in as much as precise information is available, it is used. Thus, from the side view tape, the silhouette of the automobile is measured at regular small intervals. This information is noted and subsequently used to drive one of the layout machines to place points on the model corresponding to this measured information. This work results in a planar set of points down the middle of the model. Boundaries for these surfaces are determined from the tape drawings and these are also marked on one side of the model. Because the automobile is obviously symmetric across the mid plane, only one side is modeled at a time (except for those surfaces created with templates). Once a satisfactory half-surface has been created it is mirrored to create the other side – again using frequent measurement positions.

Once these surfaces have been developed, the construction of secondary surfaces can begin. Once all relatively flat surfaces have been completed, the interfaces between them are addressed. Very thin (and therefore, flexible) blades are use to create the blended fillets between the surfaces. A useful rule of thumb is to create the larger surfaces first, before attempting to make them consistent with smaller secondary surfaces.

In this way, surfaces such as the window panels are added to the model using both the measured information from the tapes (if available), and the information generated by the positioning of those surfaces which have already been created. At some point the designer will decide that as much of the model as possible has been constructed, and that it is time to view the model in the presentation room.

The presentation room is a very large space generally with large amounts of natural and artificial light. In addition, the floor normally has large rotating disks in the floor so that a model can be rotated effortlessly to be examined from various angles and in various lighting situations. This is usually approached as something of an internal review and other designers and interested parties will show up to provide the designer with ideas and advice.

The reason for moving the model to a larger space is that in the relatively cramped studio space it is difficult to fully appreciate the shape of the car. This is because the clay modeling requires that the tape drawings be nearby. This requires a great deal of nearby wall-space. In effect, this forces the studio space to be smaller than is necessary to fully appreciate the car. For example, it is easy to appreciate that the proportions of the vehicle must be satisfactory when viewed from some distance as well a up close. Although the model is usually only kept in the presentation room for a day or so, a huge amount of information is gathered in that short time. The designer will take this information back to the studio in order to help guide further revisions of the model.

Revisions and Modifications Once this review is completed, the designer and modeler return to the studio in order to act on the information gleaned from the review in the presentation room.

By now, a more or less complete model exists and it remains to enhance it to the point where it meets with the designer's approval. Thus, the designer will work closely with the

modelers to revise the model. She might do this in conversation, by changing the tapes, or by working with the modeler directly on the model. For example, a designer will often attempt to design vertical section curves at intervals along the length of the body side. In this way, she can fine-tune the "attitude" of the automobile.

The designer will often work with tape directly on the model to convey to the modeler exactly where and how two sections are supposed to merge. It is interesting to note the degree to which the modeler contributes to this process. Although the design clearly belongs to the designer, a good designer will work with – rather than over – the modeling team. Naturally, the modeling team has a great deal of practical experience to contribute to the process. By necessity, modelers have generally spent much more time actually resolving and creating three-dimensional surfaces than have designers. Clearly, it is in the designer's interest to take advantage of as much of this experience as possible.

In order to interrogate the model's surfaces at various points, there are a number of ways to enhance the reflecting quality of the normally matte clay surface. For a quick local check, alcohol based spirits are often used. Although it evaporates reasonably quickly, it gives the surface a wet shine for a short time. With such a shine parallel strips of light in the ceiling reflect clearly on the surface below. For a more thorough inspection, dinoc film is used. Dinoc film is much like a decal. It has a light glue on one side which temporarily sticks to the clay surface when wet. Thus, it can be stuck onto the surface and air bubbles underneath can be worked out using plastic edges. Finally, this highly reflecting surface can be dried. This is especially useful when the project is being reviewed in the presentation room.

In any event, when the designer chooses to modify a surface in some way, the modeler will make use of the same tools as were used to create the surface in the first place.

Final Presentation. Once the designer is satisfied with the model, it is dressed for purposes of presenting it to management. This involves adding a significant amount of detail to the existing surfaces.

This may include creating the wheels with which the car is to be fitted, as well as adding smaller fittings, such as door handles. Of course, the model is coated with dinoc film and the windows are blacked out. An example of such a final presentation model is shown in Figure 2-32. This final picture model is generally a scale model and will be used by the management to compare different designs.

A subset (or all) of the designs will be critiqued and final picture models will be developed. In this way, management has a number of options from which to choose.

In practice, a new full scale model is normally created for this final selection stage. Generally, this model is created by three-dimensionally scanning the existing model with lasers and using a five-axis milling machine to create a new full scale model. This model is normally also considered a picture model. These final picture models are complete in all external details (see Figures 2-33 and 2-34). From these final picture models, the management will choose the design which will be manufactured – if any.

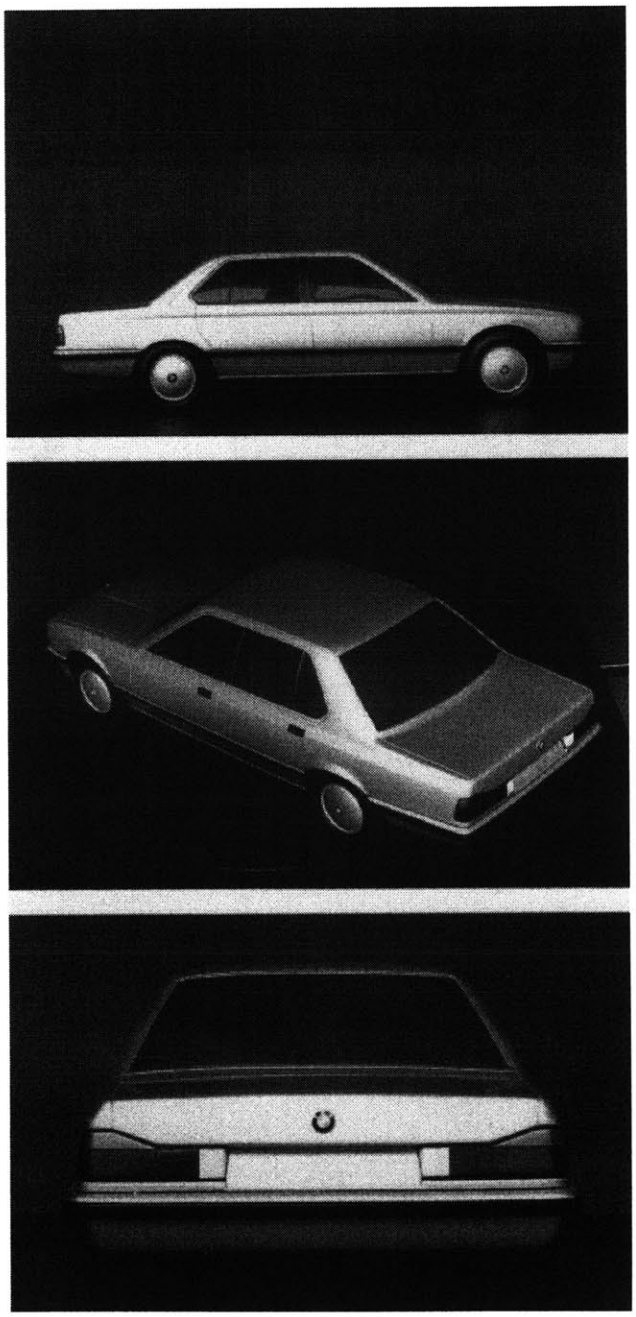


Figure 2-32: Presentation model of a BMW 7-Series. Reproduced from [Wic87].

Figure 2-33: A final picture model of a proposed BMW 7-Series. Reproduced from [Wic87].

Figure 2-34: Another final picture model of a proposed BMW 7-Series. Reproduced from [Wic87].

Feasibility Model Development

If a model is selected for production, a feasibility model must be created. This model must contain both the interior and exterior of the automobile. Again, this model's surfaces (interior and exterior) are machined and all visible details are added. This model is then turned over to those who are responsible for subsequent development of the automobile.

2.4 Preliminary Conclusions

With this overview of the automotive styling process in place, a number of preliminary conclusions may be drawn in regard to supplementing the process with computer software.

Perhaps most shocking is the fact that computer technology is so conspicuously absent from this styling process. However, a closer review of automotive styling presented here suggests some explanations for why designers have shunned such computer assistance.

Modern computer-aided design and computer aided geometric design systems provide almost no support for the way in which designs actually evolve. A design is not created in an approximately prefect state which proceeds to completion via some smoothing algorithm. Quite the opposite is, in fact, the case: A design evolves with more and more detail resolved in the course of that process. And yet, if we contemplate the capabilities of current computer aided geometric design systems, we realize that they are not at all suited to such an evolutionary design process.

In current computer systems, the basic mode of interaction is the creation of new surfaces. If a surface is not satisfactory, the most expedient way to modify it is to re-create it with slightly altered parameters. Of course, modification of surfaces via their control nets is also possible, but this method is almost uncontrollable in all but the simplest cases. Thus, there is no effective means of transforming an existing surface to a new one – thereby allowing for an evolutionary process.

In this research, I have created a system which does, in fact, support such evolution as the primary method of generating designs. Fully supporting an evolutionary design process involves much more than creating a system which truly allows the meaningful modification of existing surfaces. It must also support design at different levels of abstraction. For example, in the automotive styling process, the design actually evolves from two-dimensional approximations to a full-scale accurate model.

A useful computer system in the automotive styling domain must allow designers to design two-dimensional images as well as three-dimensional models. However, this system must also do much more. It must also permit a reasonably full range of intermediate levels between these two types of design. The next stage of this research is to specify the exact nature of this design evolution and how the system must support it.

It is important to note that rather than aiming to replace traditional automotive styling or design methods, the reason for developing this system is to create another tool which designers can choose whether or not to use. Hopefully, designers will be inclined to use the tools created here since they will not obstruct their design process the way current computer-aided design options do.

If auto stylists choose to work with such a system, there are many potential benefits. An obvious advantage is that the control data will be in a computer-ready format. This en-

ables the exchange of data between styling and computer-aided fluid dynamics departments at much earlier stages, for example. In this way, certain technical flaws in the design can be avoided in the early stages of the design evolution. This effectively allows the designer to spend more time on feasible designs. Another obvious advantage that computer-aided styling would bring with it is that laser sampling of control models would become unnecessary.

However, a much more important advantage may be derived from the fact that the designer is able to exert much more control over the evolution of the design. The ease with which she can fork the design process in order to explore design alternatives is dramatically increased if there is no clay model involved. Also, it would be possible for a designer to create an approximation of a desired effect, which the modeler could use as a guide in modifying the control model.

All in all, it is clear that the introduction of a computer system to aid in the styling of an automobile could have great benefits for designers and their designs.

2.5 Features Required in a New System

The goals of this research are to:

- Examine current aesthetically-constrained freeform surface design methodologies.
- Develop a list of features that should be present in a software implementation if it is to address freeform surface design.
- Implement a proof-of-concept system to test some of the major items in that list of required features.

Above, we have a detailed description of the freeform surface design process as it exists at BMW Ag., Munich, Germany. The process is more or less the same at other automotive design facilities.

At this point, we may describe the computational tool-set which might address the design methodologies distilled above.

2.5.1 Preliminaries

Before delving into the details of the specifications of such a system, a number of important matters must be discussed.

Interactivity

One red-herring issue that receives a great deal of attention in the CAD and CAGD literature is *interactivity*. Thus, developers of new techniques expend a great deal of effort in showing that their systems are interactive in the sense that a user can change an input and get updates of the result in a small amount of time.

On its face, it is hard to argue that this may not always be a good thing, but when discussing interactivity, it is crucial to keep in mind what exactly is being timed. For instance,

in a moderately well implemented Non-Uniform Rational B-Spline (NURBS) system running on a moderately fast computer, a user should be able to drag a control point and view updates on the screen without perceiving that there is any lag in displaying the updates. Clearly, all else being equal, a system which can accomplish such updating in real time or at least at interactive rates will always seem better than one that cannot.

However, a system that is interactive at every step is not necessarily better than one that is not in any general sense. For instance, consider a user who needs to modify a surface in that well-implemented NURBS modeling system. If that user needs to modify the surface by simply moving one control point, the interactive system will always seem better. Yet, if the user needs to move every control point so that the resulting surface looks like a nice patch on a dome, then it may be far more desirable to have a system that can accept some type of definition of a dome and then, at the user's request, modify the surface to match that definition. While this seems trite, it is an often forgotten fact.

The user who must move every control point individually may spend hours reshaping the resulting surface while the user of the other system would only spend a fraction of that time. Of course, the dome tool on the new system, may actually need to process the input in a non-interactive way, but if the total time is smaller, then the user will be getting more work done.

The obvious conclusion here is that interactivity is not nearly the panacea that it is made out to be. The real measure of the speed of a system is not the speed of interaction with it, but rather the amount of time it takes to get a given amount of work done. Thus, for example, a very fast system which does not support the tools necessary to performing some task, will always be inferior to any system, no matter how slow, which actually enables the work to be done!

Another important observation on this subject is that the goal of interactivity may have arisen when computers were so slow (relative to today's systems) that simply being able to move control points at near interactive speeds was a dream in itself. But one is forced to observe that many of the very clever techniques developed to achieve interactivity are strictly far less important today since even a naive implementation of a NURBS surface can remain interactive during editing (of course, at some point the surface may become sufficiently large that this no longer holds).

Local Control

Local control is itself another important issue that is often cited as a major benefit of using NURBS over other surface representations, such as the finite element system implemented in this research.

Again, a definition of terms is important here. Local control defines a property of B-Splines (and some other representations as well) that captures the fact that when the user drags a single control point, only a well-defined subregion of the surface near that control point will actually be deformed. A clear advantage of this property is that in the implementation of a B-Spline surface editing tool, only that portion of the surface that is actually changing should be recomputed. Thus, a surface may become arbitrarily large (in terms of the number of control points defining it) and the time taken to update the surface after a control point drag will remain constant. This seems to be a very desirable

property. In fact, it was extremely important in the early days of CAD tools when local control allowed a user to perceive much greater responsiveness in a CAD system.

But local control comes at a price. In fact, the price is simply that moving a single control point is *not* distributed over the surface. Thus, for the designer, it becomes much more tedious to modify a surface in some global way (to create a dome for example). Thus, the user of B-Spline system is encouraged to manipulate a surface using its mathematical controls, the control points, even though these points are neither on the surface itself nor are they particularly effective when it comes to modifying surfaces globally.

As a consequence, a great deal of research has gone into developing algorithms which will modify sets of control points in order to achieve some goal – perhaps enforcing concavity across a surface to the greatest extent possible. Stepping back, this seems to be a peculiar goal as these techniques are really just embedding non-local control back into the surface.

Local control is problematic from another vantage point as well. When viewed from the perspective of a designer, local control implies that she could choose an arbitrary sub-region on a surface and modify that subregion without affecting the larger surface. The designer's notion of local control is, therefore, not supported by the mathematical notion of local control. In fact, for a given B-Spline surface, the regions in which local control is possible are predefined and static. Specifically, for a B-Spline surface of degree 3 in both parameter directions, and n control points in one direction and m in the other, there are exactly $(n-3)(m-3)$ distinct sub-regions (spans) on the surface. And, if the user moves a particular control point, up to 16 ($(degree+1)$ in each direction) of these spans will need to be updated (it could be fewer than 16 depending on how near the control point is to the surface boundary). Local control comes from the fact that 16 is a constant (determined solely by the degree of the surface).

The user has no immediate means of defining their own region of local control. Thus, local control, as the mathematician understands it, is purely an implementation detail and is of almost no value to the user unless their local control needs happen to coincide with those inherent in the representation.

However, the local control required by designers is more than the mathematical concept. Specifically, for the designer, local control involves the ability to not only specify an arbitrary sub-region on a surface, but also to make arbitrarily complex modifications to the surface in that sub-region – something that is completely unsupported in the mathematical implementation of local control. To make this concrete, a designer may begin with a very simple surface, say, a bicubic with 16 control points. If this surface were to represent a car door, she might push and pull control points until the surface had the general shape she required. She might then decide that she needs to model the door-handle area of the door. What she intuitively wants to do is mark the surface to signify where the handle will be (i.e. define a sub-region of the surface) and to lock the surface outside that region. She will then want to edit that sub-region directly. In the B-Spline representation, even if she could define a sub-region on the surface (which she cannot because it is defined as a single span already), there would be no control points available which would only affect that region.

Thus, if the B-Spline system were to support such design, it would either have to insert control points into the surface so that the subregion has a sufficient number of control points or it would have to split that surface into 9 smaller surfaces. In either approach, due to the fact that B-Spline surfaces are defined over regular grids, the control point density

required in the sub-region would be propagated to the surface outside the region – thus creating control points in regions where they are not benefiting the user (in fact, they are now definitely a hindrance due to the local control property itself, as the larger surface now becomes virtually uneditable due to its complexity). In the second approach, the system would have to ensure that the surface-surface joins remain stitched during subsequent editing. In short, what was a conceptually simple demand must now be implemented in a very contrived way.

2.5.2 Application Requirements

The overarching goal of a new system to aid in the design of freeform surfaces is that it must let the user design such surfaces more quickly than current CAD tools and it must allow the user to fully explore designs within the system – two requirements which are essentially unsupported in current systems.

Having covered a considerable amount of the context in which this application was developed, we are now in a position to set down some of the requirements for such a system.

- Sketch Tool: It may suffice to take advantage of already existing sketching and painting tools (such as Quantel's PaintBox, Adobe's Photoshop or even the Gimp). However, given the specific characteristics of what is being sketched, it may be possible to create a more targeted paint system for sketching during the design process.
- Tape Tool: A user interface for the creation of tape drawings which, where possible, could ensure consistency between the various tape drawings as well as with package data. It is worth exploring a new physical interface for this tool. Specifically, designers take advantage of the fact that the tape has some lateral resistance to bending in order to help them create smoother curves. Perhaps a dial-box interface can be developed or a full-blown three-dimensional virtual reality interface.
- Tape-To-Model Tool: Given the output from the Tape Tool or scanned tape drawings, it would be useful to have a tool that enabled the designer to automatically (or almost automatically) create a starting model from which to work.
- Sketch Projection Tool: Given a rough model (perhaps the output of the previous step), we would like to be able to project sketches onto it so as to get a sense of the three-dimensional form. Perhaps, a library of basic car shapes would be collected so that designers could project their sketches long before they have created tape drawings.
- Three-Dimensional Tools: This listing is probably not exhaustive, yet it does provide a very good start:
  - Section Sweep Tool: Given an arbitrary curve-on-surface, the user places section planes along the curve and edits a cross-section curve within each section plane. The tool will create a sweep surface along the curve-on-surface which interpolates the defined cross-sections and then integrate resulting sweep surface into the original surface.

  - Dragging Sweep Tool: Similar to the Section Sweep Tool except that the section plane is dragged along the curve-on-surface as the user interactively modifies the cross-section during the drag.
  - Shape-from-Shading Tool: A screen capture is made under known lighting conditions and the user then edits that captured image using the Sketch Tool or any image processing system. Once the editing is finished, the system determines how the original image has been modified and attempts to deform the underlying surface so that it produces the sketched shading patterns under known lighting conditions.
  - Define Region Tool: Support local control from the perspective of the designer.
  - Create And Embed Curve-On-Surface Tool: Many of the tools rely on being able to define a curve on a surface.
  - Move Point-On-Surface Tool: For a point on the surface, let the user move it as she desires. This should be extended to let her directly manipulate the tangent plane at a point, etc.
  - Move Point-On-Surface Along Normal Tool: Similar to the Move Point-On-Surface Tool except that points are moved along the direction of the surface normal at the point.

- High Quality Renderer: Clearly, it would be useful to be able to produce high quality renderings of surfaces generated in this system.

- Virtual Reality Interfaces: Almost any of the tools in this requirements list could be converted to having a virtual reality interface – probably with great benefits for designers. Some specific ideas:

  - Tape Tool: An interface to this tool enabling the user to gain advantages similar to those to be had from using physical tape, seems well within reach using current technology and would be a very effective design tool.
  - Tape-To-Model Tool: To the extent that this tool requires manual intervention, it would probably be much easier for a user to guide the system effectively given a proper three-dimensional view of the model.
  - Sketch Projection Tool: If nothing else, being able to view the models with the projected sketches in a virtual reality environment is desirable. It should also be possible to enable a designer to interactively place the textures in such an environment.
  - Dragging Sweep Tool: A completely new interface for this tool seems possible and certainly desirable.

This is obviously a fairly long list of features that would be required – and certainly not a complete list. Nevertheless, in a system that properly addresses each of these requirements, a designer would be able to design freeform surfaces in a satisfactory way.

In addition to this feature list, we can say certain things about the underlying representation that would make the system even more effective: The properties of the representation should include:

- Supporting a designer's definition of local control: The designer should be able to define an arbitrary closed boundary on a surface and define how the portions of the surface on either side of the boundary will interact.
- Supporting global control: The designer must be able to propagate edits of the surface must be able to be propagated across the surface to the extent that locally controlled regions allow.
- Supporting implementation of *level of detail*: Because designers require wide latitude (and not high quality) early in the design process and less latitude (and high quality) later in that process, it would be desirable if the underlying surface representation could operate in both of these capacities.

2.6 Conclusions

At this point, this research has yielded a description of the freeform surface design process, along with the requirements for a system to support it.

The remainder of this research is to implement a proof-of-concept system incorporating a number of these requirements. Specifically, the work concentrates on the three-dimensional tools described above. In addition to implementing proofs-of-concept for each of the three-dimensional tools described above, I embedded a high quality renderer (Blue Moon Rendering Tools [Gri00]) into the system along with a few reflection-line shaders for surface inspection.

While extensive testing of the implementations as they now exist will undoubtedly lead to enhancements and modifications, these proofs-of-concept succeed in demonstrating the value of reassessing current CAGD approaches when addressing the complex problem of three-dimensional freeform surface design. In fact, not only do they demonstrate the value of reassessment, but they also point out new and fruitful directions for future implementations targeted at freeform surface design.

Chapter 3

Fundamental Theory

By way of establishing the technical context in which this work has been undertaken, it is useful to begin by describing some of the fundamental theory which forms the basis of this work. Although this research will draw on a number of research areas, it is primarily concerned with three specific disciplines. The most fundamental of these is the Mathematical field of differential geometry which addresses the analysis of surfaces. Next, is the research field of computer aided geometric design (CAGD) itself. Finally, Finite Element Methods (FEMs) figure prominently in this work, and so, the basic techniques and issues are set down here.

3.1 Differential Geometry

The field of computer aided geometric design is, itself, rooted in Differential Geometry. While it is certainly beyond the scope of this work to treat differential geometry in anything approaching a thorough way, this section will provide a brief overview of the field – especially as it most affects CAGD. This presentation is drawn from a number of excellent references which provide a thorough and complete treatment of the subject [Kre91, Mae95, Str88].

3.1.1 Explicit, Implicit and Parametric Formulations

There are three principal ways of defining curves and surfaces. They are the *implicit*, *explicit* and *parametric* definitions of a curve or surface. Each has certain advantages and disadvantages. It is the parametric formulation which is most used in the CAGD community.

Implicit equations are ones in which the variables are embedded. Thus the implicit equation of a circle in the xy-plane of radius a and centered at the origin is

$$x^2 + y^2 = a^2$$

and, not surprisingly, that of a sphere is

$$x^2 + y^2 + z^2 = a^2.$$

This is a reasonably useful class of equations for representing curves and surfaces. More generally, we can consider implicit equations to be of the form $f(x, y) = 0$ or $f(x, y, z) = 0$.

Curves and surfaces defined explicitly are slightly more difficult to manipulate although they are somewhat easier to visualize quickly. In this type of equation, one of the variables is given as a function of the others. Thus, in this form, a circle would be written as

$$\begin{aligned} y &= \pm\sqrt{a^2 - x^2} \\ or\ y &= f(x) \end{aligned}$$

while the sphere would be written

$$\begin{aligned} z &= \pm\sqrt{a^2 - x^2 - y^2} \\ or\ z &= f(x, y). \end{aligned}$$

These simple examples suggest one of the main drawbacks of the representation. The $\pm$ is necessary in both of these cases because the implicit equation is sufficiently complex. Although implicit equations are not effortlessly stored in computer memory, it is pretty easy to imagine that storing the explicit form of the circle would be rather difficult and the required display routines would be littered with special cases.

Finally, the most useful representation from the point of view of the CAGD community is the parametric form of a curve or surface. In this form, a curve or surface is said to have a parametric domain. In turn each location on the curve or surface is defined as function of a parameter (or parametric coordinate). Thus, the circle could be written

$$\begin{aligned} \mathbf{c}(t) &= [a\cos t, a\sin t], \quad 0 \le t < 2\pi \\ or\ \mathbf{c}(t) &= [x(t), y(t)], \quad 0 \le t < 2\pi \end{aligned}$$

and the sphere

$$\begin{aligned} \mathbf{r}(u,v) &= [a\cos v\cos u, a\cos v\sin u, a\sin v], \quad 0 \le u < 2\pi,\ -\tfrac{\pi}{2} \le v < \tfrac{\pi}{2} \\ or\ \mathbf{r}(u,v) &= [x(u,v), y(u,v), z(u,v)], \quad 0 \le u < 2\pi,\ -\tfrac{\pi}{2} \le v < \tfrac{\pi}{2} \end{aligned}$$

This is a very powerful representation. However, it is not without drawbacks. Most pronounced among these is that a single curve (or surface) can be reparametrized in an infinite number of ways (without changing its shape). For example, the parametric circle above can be *reparametrized* to be

$$\mathbf{c}(t) = [a\cos 2t, a\sin 2t],\ 0 \le t < \pi.$$

While this may appear to be a very small and insignificant change, the change makes a considerable difference in the curve's properties. It is easy to sense why the difference is important by considering the derivative of the curve. In the first case, the parametric derivative is given by

$$\dot{\mathbf{c}}(t) = [-a\sin t, a\cos t],\ 0 \le t < 2\pi$$

while in the second case the corresponding derivative is given by

$$\dot{\mathbf{c}}(t) = [-2a\sin 2t, 2a\cos 2t],\ 0 \le t < \pi.$$

These derivatives define vectors which are consistent in the sense that they both travel along the same line at a given point on the curve. However, these vectors are **not** of the same length. At corresponding points on the circles, the derivative vector of the second circle is always twice as long as that of the first. The only parameterization which produces the true derivatives is the arc length parameterization. However, although we do know that arc length is $2\pi a$ in this case, it is clear that before we have fully defined a curve, we cannot know the arc length in a general case.

As a result of this difficulty, and the obvious importance of derivatives, researchers have developed a new way of looking at them. Derivatives are obviously crucial to determining the degree of continuity between curves or surfaces. In the CAGD field, the notion of *visual* or *geometric* continuity has taken on great importance. This new measure is concerned with the unit derivatives and is thus independent of parameterization. Such visual continuity offers certain advantages over normal continuity measures.

3.1.2 First Fundamental Form

This *First Fundamental Form* of a surface is derived from the arc-length of a curve on a surface. Given a surface $\mathbf{r} = \mathbf{r}(u, v)$ defined in the uv-parameter space, we can say that $\varphi(u, v) = 0$ defines a curve on that surface. We can then rewrite that curve as $\mathbf{r} = \mathbf{r}(u(t), v(t))$ where t is a new parameter. The vector $d\mathbf{r}/dt$, or $\dot{\mathbf{r}}$, at a point on the surface is clearly tangent to the surface at that point. The vector $\dot{\mathbf{r}}$ is given by

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{r}_u\dot{u} + \mathbf{r}_v\dot{v}, \\ or\ d\mathbf{r} &= \mathbf{r}_u du + \mathbf{r}_v dv \end{aligned} \tag{3.1}$$

in differential notation. The distance between two points on the curve $\mathbf{r}(u(t), v(t))$ at parameter values t_0 and t_1, respectively, is found by integrating the square root of

$$ds^2 = d\mathbf{r} \cdot d\mathbf{r} \tag{3.2}$$

from t_0 to t_1. But substituting (3.1) into (3.2) gives

$$ds^2 = (\mathbf{r}_u du + \mathbf{r}_v dv) \cdot (\mathbf{r}_u du + \mathbf{r}_v dv)$$

which in turn gives

$$ds^2 = E du^2 + 2F du dv + G dv^2$$

where

$$E = \mathbf{r}_u \cdot \mathbf{r}_u,\ F = \mathbf{r}_u \cdot \mathbf{r}_v,\ G = \mathbf{r}_v \cdot \mathbf{r}_v. \tag{3.3}$$

So, the First Fundamental Form is given by

$$I = E du^2 + 2F du dv + G dv^2. \tag{3.4}$$

It is sometimes referred to as the *metric tensor* or the *fundamental tensor*. In the tensor formulation, the coefficients make up a 2×2 symmetric matrix which is, in fact, the tensor. Thus, the First Fundamental Form, given in terms of the metric tensor (or first fundamental matrix) is

$$I = \begin{bmatrix} du & dv \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix} \tag{3.5}$$

where

$$g_{11} = E,\ g_{12} = g_{21} = F \text{ and } g_{22} = G \tag{3.6}$$

Frequently, the metric tensor is denoted by $\mathbf{G}$.

The Fundamental Form is important because it provides a measure of the arc length of a curve on a surface. Thus, it could give the ideal reparametrization in arc length as discussed above. However, it also provides a way of interrogating the surface for certain properties. For example, it is possible to formulate an equation for a differential area (dA) on the surface as follows

$$\begin{aligned} dA &= |\mathbf{r}_v du \times \mathbf{r}_u dv| \\ &= \sqrt{\mathbf{r}_u{}^2 \mathbf{r}_v{}^2 - (\mathbf{r}_u \cdot \mathbf{r}_v)^2}\, du dv \\ &= \sqrt{\det(\mathbf{G})}\, du dv \end{aligned}$$

which, when integrated over a u, v-domain, gives the surface area on the surface corresponding to the domain. Thus,

$$A = \int_{v_0}^{v_1} \int_{u_0}^{u_1} \sqrt{\det(\mathbf{G})}\, du dv \tag{3.7}$$

gives the surface area in the domain $u_0 \leq u \leq u_1$ and $v_0 \leq v \leq v_1$. This form is also extremely useful in conjunction with the Second Fundamental Form in order to determine higher order surface properties.

3.1.3 Second Fundamental Form

While the First Fundamental Form was reached by considering the arc length of a curve $\varphi(u, v) = \mathbf{r}(u(t), v(t))$, the Second Fundamental Form is derived by considering the curvature vector $\mathbf{k}$ of that curve on the surface at an arbitrary point $\mathbf{P}$. Let $\mathbf{t}$ be the unit tangent of the curve at $\mathbf{P}$. Then, $\mathbf{k}$ is the vector $d\mathbf{t}/ds$ where ds is just the differential arc length at that point. But $d\mathbf{t}/ds$ is also given by $\kappa\mathbf{n}$ where $\mathbf{n}$ is the principal unit normal to the curve at $\mathbf{P}$. Clearly, $\kappa\mathbf{n}$ can be decomposed into components normal and tangent to the surface in which the curve lies. Thus, we have

$$\mathbf{k} = \frac{d\mathbf{t}}{ds} = \kappa\mathbf{n} = \mathbf{k}_n + \mathbf{k}_g \tag{3.8}$$

where $\mathbf{k}_n$ is in the direction normal to the surface and $\mathbf{k}_g$ is tangent to the surface. The vector $\mathbf{k}_n$ is defined to be the *normal curvature vector* and $\mathbf{k}_g$ is defined to be the *geodesic curvature vector* or *tangential curvature vector.* So, if a unit surface normal is called $\mathbf{N}$, then

$$\mathbf{k}_n = \kappa_n \mathbf{N} \tag{3.9}$$

where κ_n is defined to be the normal curvature of the surface at $\mathbf{P}$ moving in the direction of $\mathbf{t}$. Even though κ_n is derived in terms of a curve on the surface passing through $\mathbf{P}$ in the direction $\mathbf{t}$, Meusnier gives a theorem in which he proves the following statement: *All curves through* $\mathbf{P}$*, tangent to the same direction, have the same normal curvature vector.* Thus, $\mathbf{k}_n$ depends only on the vector $\mathbf{t}$ at $\mathbf{P}$ and the sign of κ_n depends on how we have oriented our surface (i.e. the choice of unit normal field N).

Clearly, $\mathbf{N} \cdot \mathbf{t} = 0$, and thus, by differentiating this along the curve φ we have that

$$\frac{d\mathbf{t}}{ds} \cdot \mathbf{N} + \mathbf{t} \cdot \frac{d\mathbf{N}}{ds} = 0$$

by the product rule. But then we can write

$$\frac{d\mathbf{t}}{ds} \cdot \mathbf{N} = -\mathbf{t} \cdot \frac{d\mathbf{N}}{ds} = -\frac{d\mathbf{r}}{ds} \cdot \frac{d\mathbf{N}}{ds}. \tag{3.10}$$

But, recalling Equation (3.2), we can then write (because $\kappa_n = \mathbf{N} \cdot d\mathbf{t}/ds$)

$$\kappa_n = -\frac{d\mathbf{r} \cdot d\mathbf{N}}{d\mathbf{r} \cdot d\mathbf{r}} \tag{3.11}$$

$$= -\frac{d\mathbf{r} \cdot d\mathbf{N}}{I} \tag{3.12}$$

where I is the First Fundamental Form. Thus, it remains to evaluate $d\mathbf{r} \cdot d\mathbf{N}$ and this is done as follows

$$\begin{aligned} d\mathbf{r} &= \mathbf{r}_u du + \mathbf{r}_v dv \\ d\mathbf{N} &= \mathbf{N}_u du + \mathbf{N}_v dv \\ \Rightarrow -d\mathbf{r} \cdot d\mathbf{N} &= -(\mathbf{r}_u \cdot \mathbf{N}_u)du^2 - (\mathbf{r}_u \cdot \mathbf{N}_v + \mathbf{r}_v \cdot \mathbf{N}_u)dudv - (\mathbf{r}_v \cdot \mathbf{N}_v)dv^2. \end{aligned} \tag{3.13}$$

But

$$\begin{aligned} \frac{\partial}{\partial u}(\mathbf{r}_u \cdot \mathbf{N}) &= \mathbf{r}_{uu} \cdot \mathbf{N} + \mathbf{r}_u \cdot \mathbf{N}_u = 0 \\ \Rightarrow \mathbf{r}_{uu} \cdot \mathbf{N} &= -\mathbf{r}_u \cdot \mathbf{N}_u \end{aligned}$$

and similarly,

$$\begin{aligned} \mathbf{r}_{vv} \cdot \mathbf{N} &= -\mathbf{r}_v \cdot \mathbf{N}_v \\ \textit{and } \mathbf{r}_{uv} \cdot \mathbf{N} &= -\mathbf{r}_u \cdot \mathbf{N}_v \\ &= -\mathbf{r}_v \cdot \mathbf{N}_u. \end{aligned}$$

So, finally, we arrive at the Second Fundamental Form

$$II = Ldu^2 + 2Mdudv + Ndv^2 \tag{3.14}$$

where

$$L = \mathbf{r}_{uu} \cdot \mathbf{N},\ M = \mathbf{r}_{uv} \cdot \mathbf{N}\ and\ N = \mathbf{r}_{vv} \cdot \mathbf{N}. \tag{3.15}$$

Thus, we have an equation for the normal curvature of the surface in the direction corresponding to the tangent to the curve at P is $\kappa_n = \frac{II}{I}$.

As in the case of the First Fundamental Form, this is often rewritten in tensor form

$$II = \begin{bmatrix} du & dv \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix} \tag{3.16}$$

where

$$b_{11} = L,\ b_{12} = b_{21} = M\ and\ b_{22} = N$$

and where this second fundamental matrix is often called **B**.

3.1.4 Curve and Surface Analysis

Together, the First and Second Fundamental Forms of a surface provide the basic information for a great deal of surface interrogation. We are now in a position to specify various other important curvature measures on the surface. By defining the *Gauss* curvature K and the *mean* curvature H as follows

$$\begin{aligned} K &= \frac{LN - M^2}{EG - F^2} \\ H &= \frac{EN + GL - 2FM}{2(EG - F^2)} \end{aligned}$$

we can write down the *principal* curvatures. These are given by

$$\kappa_{max} = H + \sqrt{H^2 - K} \tag{3.17}$$

$$\kappa_{min} = H - \sqrt{H^2 - K}. \tag{3.18}$$

Thus, we can see that

$$K = \kappa_{max}\kappa_{min} \tag{3.19}$$

$$and\ H = \frac{\kappa_{max} + \kappa_{min}}{2} \tag{3.20}$$

With these derivations and definitions in hand, we now are in a position to analyze many important properties of a free-form parametric surface.

3.2 Parametric Surfaces

Many of the earliest efforts to represent surfaces in a computational medium were driven by the desire to machine parts using numerically controlled tools. To a great degree, this remains a fundamental motivation today. By means of such NC tools, many different machining tools can be controlled. An NC model is little more than a path which the tool should follow in order to produce the surface or general form that the model represents. Obviously, the nature of the path information will change depending on the machine, but NC methods standardized this protocol. Obviously, many industries were very interested in these control technologies and it was realized that significant time saving could be had if the surfaces could actually be developed directly in a computerized medium rather than be "scanned" into the computer system. Early attempts at solving this problem were devised by Bézier [Béz66, Béz67, Béz68, Béz74] and de Casteljau who simultaneously developed what is known as the Bézier curve. More or less simultaneously, Coons and Gordon developed what has become known as the Coons patch [Coo67, GW72, GR74]. Also at more or less the same time, but at Boeing, Ferguson developed the Ferguson or Cardinal spline [Fer64].

Of these, the two representations in most common use are the Bézier and B-Spline representations.

3.2.1 The Bézier Representation

Borrowing terminology from [Far90], Bézier curves are most often expressed in terms of Bernstein polynomials. A Bernstein polynomial has the form:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{3.21}$$

where

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \le i \le n \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

and n is the degree of the polynomial. An important property of Bernstein polynomials is that they form a *partition of unity* which simply means that for a given degree n and a given valid t, the $n+1$ Bernstein polynomials always sum to 1:

$$\sum_{j=0}^{n} B_j^n(t) \equiv 1. \tag{3.23}$$

We can now write down a formula for a Bézier curve as follows:

$$\mathbf{b}^n(t) = \sum_{i=0}^{r} \mathbf{b}_i^{n-r}(t) B_i^r(t). \tag{3.24}$$

where the $\mathbf{b}_i^r$ are known as the de Casteljau points and are given by

$$\mathbf{b}_i^r(t) = \sum_{j=i}^{i+r} \mathbf{b}_j \left[(1-t)B_{j-1}^{r-1}(t) + tB_{j-i-1}^{r-1}(t)\right]$$

3.2.2 The B-Spline Representation

Where Bézier curves are defined using the Bernstein polynomials as basis functions, B-Splines are defined using B-Spline basis functions. Using terminology borrowed from [PT97] the i-th B-Spline basis function of degree p is given by the recursive formula

$$N_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} N_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1}^{p-1}(u) \tag{3.25}$$

where u is a parameter value and where

$$N_l^0(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{3.26}$$

A B-Spline curve is defined using a sequence of points known as a control polygon. A curve that is to have L spans will have $L + p$ points in the control polygon. In addition to the control polygon, a B-Spline curve definition requires a *knot vector* $\{u_0, \ldots, u_{L+2p+1}\}$. Finally, a B-Spline curve can be written as follows:

$$\mathbf{C}(u) = \sum_{i=0}^{n} \mathbf{P}_i N_i^p(u) \tag{3.27}$$

where n is the number of control points for the curve.

B-Spline curves and surfaces have become the de facto standard curve and surface representation schemes in the CAD and CAGD fields. In addition to Farin's excellent reference [Far90], Piegl and Tiller have also produced a superb book on NURBS (Non-Uniform Rational B-Splines) [PT97]. In fact, Piegl and Tiller originally developed the rational form of the B-Spline [PT87].

3.3 Finite Element Method

Another area of fundamental research which has a bearing on this research is that of finite element analysis. As already suggested, many of the techniques employed to embed shape information into the surface itself are based on the finite element method. In fact, the work that most directly precedes this research was also based on such an approach. Celniker's work on the *ShapeWright* functional was predicated upon the ultimate finite element solution of the system of equations set up for the surface [Cel90]. It is therefore appropriate to provide some background in this area.

Finite Element Methods provide an essential tool for engineering analysis in a wide range of ways. Application domains include fluid flow analysis (including aerodynamic analysis) and structural analysis to name two. The study of these techniques is and has been a

thriving area of research for at least forty years. As a consequence, there are excellent reference works on the subject ([Bat82, ZT89] for example). There are a number of ways to derive the fundamental finite element equations but this account will begin from variational principles.

3.3.1 Variational Formulation

Many textbooks derive the finite element equilibrium equations from the *principle of virtual displacements*. This principle holds that a body's equilibrium requires that for any small virtual displacement consistent with the boundary conditions imposed on the body, the total internal virtual work is equal to the total external virtual work.

Another starting point is a variational statement of a problem: find a shape that minimizes some function over the object. Specifically, for a surface the general form of such a statement is

$$I(S) = \int_S f(u,v)dudv \tag{3.28}$$

where S is a surface parametrized by u and v. The function $f(u,v)$ is often referred to as the energy functional. To make this concrete, Celniker's *ShapeWright* variational statement is

$$I_{surface}(w) = \int_{surface} \left(\begin{array}{l} (\alpha_{11}\mathbf{w}_u^2 + 2\alpha_{12}\mathbf{w}_u\mathbf{w}_v + \alpha_{22}\mathbf{w}_v^2) \\ (\beta_{11}\mathbf{w}_{uu}^2 + 2\beta_{12}\mathbf{w}_{uv}^2 + \beta_{22}\mathbf{w}_{vv}^2) \end{array} + \right) - 2\mathbf{f}\mathbf{w}dudv \tag{3.29}$$

where $\mathbf{w}$ is the shape function, α and β are weighting terms specific to Celniker's formulation [Cel90]. The key to the finite element method is to consider $\mathbf{w}$ as a weighted sum of discrete shape functions φ_i as follows

$$\mathbf{w}(u,v) \approx \mathbf{w}^h(u,v) = \sum_i \mathbf{x}_i\varphi_i(u,v) \tag{3.30}$$

where $\mathbf{w}^h$ is that weighted sum and the $\mathbf{x}_i$ are the unknown weights.

This is a very brief overview of the finite element method and much more information is available in [Bat82, ZT89, Cel90]. However, in finite element implementations, the φ_i are known as finite element interpolants. For instance, in the proof-of-concept system developed here, surfaces are defined over triangulated meshes and the finite element interpolants used are thus triangular as well.

3.3.2 Elements and their Interpolation Functions

Triangular elements are often defined in terms of a barycentric coordinate system over the triangle. Consider a point (u,v) in the domain of the surface in question. Barycentric coordinates L_1, L_2 and L_3 for that location in terms of the vertices (u_1,v_1), (u_2,v_2) and

(u_3, v_3) are given by

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix}. \tag{3.31}$$

This barycentric system is generally preferred as it is symmetric with respect to each of the three vertices of the triangle.

Shape functions are constructed as a weighted sum of base functions over the triangle. Celniker employed 9-DOF and 12-DOF elements in his research [Cel90]. I found that the 9-DOF element he chose was unsatisfactory and I added Specht's 9-DOF element (as described in [ZT89]) to my system.

For all of the 9-DOF interpolants that are pertinent here, the nine degrees of freedom are the position w and derivatives w_u and w_v at each vertex. The 12-DOF element adds a mid-edge normal derivative w_n on each edge.

3.3.3 Solving the Finite Element Equations

Given a finite element interpolant function, each element is defined in terms of the 9 or 12 degrees-of-freedom for that element. Thus, clearly, adjacent elements will have some degrees-of-freedom in common. Further, some of these degrees-of-freedom must be known or the system will be under-constrained. Thus, Equation 3.29 becomes a large system of equations which can be integrated numerically using Gaussian Quadrature and then solved using standard matrix algebra techniques [Dem93]: in this implementation, standard Gaussian elimination with partial pivoting was used.

Chapter 4

Previous Work

Building on differential geometry, parametric surface theory as well as the finite element method described in the previous chapter, this chapter provides an overview of some of the most significant and relevant work that has directly impacted this research. The first section contains a summary of the major efforts which have been made in order to address the user-load problem. This section concludes with a review of some of the ways researchers have attempted to control shape by subjecting the surface formulation to a minimization of some quantity – usually related to curvature (and thus, the potential energy) of the surface. In the next section, the scattered data interpolation is discussed. Many researchers have been interested in ways of imparting information to the user about a surface so that better decisions might be made regarding how that surface could be improved. This issue of surface quality is also discussed in this chapter. Finally, the use of triangular elements and their interpolants is reviewed.

4.1 The User Load Problem

As mentioned in the previous chapter, a great deal of work has been done to address the enormous user-load involved in the definition of a surface while less has been done to address the user load problems associated with editing surfaces. This has been approached in many ways, and this section aims to enumerate the main themes of such research.

4.1.1 Problem Instances

This user load problem crops up in many different design environments. When working on Frank Gehry's fish sculpture for the city of Barcelona, I attempted to replace the existing small physical control model with a computer generated one. Thus, naturally, my efforts had to meet with the approval of the architect himself. It turned out to be easier to restart a skinning process from scratch, than to attempt to manipulate control points in order to include each change Mr. Gehry requested. Obviously, this was not a very efficient process by any measure and similar problems have been reported in many other fields. For example, Andersson et al. [AAB+87] comment on the difficulties experienced at Volvo in controlling global convexity of exterior (Class I) panels. While their paper does present a means of addressing the problem itself, based on their calculations of modeling times, it would take

an experienced system operator approximately 170 hours to develop a surface description for the hood of one of their cars. Clearly, there are many problems here and it is worth looking at some attempts (including theirs) that have been made to address them.

4.1.2 Working in a Virtual Environment

One of the earlier attempts to address the problem focused on improving the man-machine interface rather than modifying the surface representation itself. This idea is described by Clark [Cla76] and he proposed using some new hardware. Essentially, there were two pieces of hardware necessary in addition to the CPU itself. He made use of Sutherland's head-mounted display in conjunction with a new mechanical wand which permitted the user to select and move control points. Thus, the user is effectively "embedded" in the virtual 3-space and in this space is able to modify the surface. At the time the work was done, it probably seemed reasonable to anticipate that soon, every CAGD station would be able to support such a virtual environment. Indeed, this may yet prove to be true. However, this expectation was not to be realized quickly and many other researchers turned to the underlying representation schemes. Although this idea has not (to my knowledge) been explicitly tested with surface quality concerns in mind, it is not clear how such a virtual environment would really help the user decide *how* to modify the control points. It is only clear that once the user had determined which control point to move and how to move it, it might be easier to accomplish such a move in a virtual modeling system.

4.1.3 Sampling Interfaces

One approach to reducing the amount of data that the user must supply has been to let implementations infer as much as possible from potentially sparse or imprecise input. Jensen, Petersen and Watkins proposed just such a scheme for generating G^2 curves from sketched user input [JPW91]. Thus, pointer positions are sampled as the user drags out a desired curve. Fang and Gossard improved on this by providing an algorithm that produces a curve from an unorganized data set [FG95]. Jensen et al. realized that the most common way surfaces are created in CAGD systems is by constructing characteristic curves, so they were effectively addressing the user load problem by making it easier create curves.

4.1.4 Beginning with a Predefined Shape

Another interesting line of inquiry has been to pursue the benefits gained by letting the user choose a beginning surface from which the final shape is derived. Thus, much of the initial user load is avoided. This method has been especially useful in areas where overall shapes do not fundamentally change from one project to the next. For example, Rogers et al. [RSR83] describe a method for simplifying the process of designing a ship hull. Their system reduces the user load by supplying a starting hull which the user then modifies. In this way, the user is spared the bother of having to create a new hull from scratch with each new project. This has certain advantages, especially within a field like ship hull design, however, it is possible that the general application of this scheme might have the cumulative effect of discouraging innovation in the fundamental shapes themselves. Nevertheless, this does not seem to be an enormous disadvantage when the scheme is applied at the early

stages of ship hull design. It also assumes powerful surface editing tools which, in general, do not exist.

Beeker [Bee86] suggests another related approach. A distinction is made between solid models and surface models. The process he advocates is to use a solid modeler to develop a rough "sketch" of the surface (modeled as 4-sided) and then, using another system, to convert these polygonal entities to Bernstein-Bézier patches. Finally, using yet another system, these patches can then be modified as Bernstein-Bézier patches. The particular system in question only produces patches with C^1 continuity, but the author rightly points out that there is no real reason why the system model could not be generalized to produce surfaces with higher degrees of continuity.

This latter idea of manipulating a model using different metaphors at different times is very intriguing and I feel that there may be a great deal yet to be gleaned from such an approach. Unfortunately, the specific system presented in Beeker's paper has a certain one-way quality, but it would be interesting to investigate the possibility of a system which allowed the seamless changing of representation. In my research, I address this issue.

Recent work has built on this initial idea of Beeker's. Shin and Shin, and Kuo developed systems that reconstruct three-dimensional solid models from orthographic views [SS97, Kuo98]. Raviv and Elber take this further and actually develop a nice sculpting tool based upon a similar projection idea [RE00].

4.1.5 Swept Surfaces

Another important area of research into the user load problem has focused on sweeping surfaces. Coquillart [Coq87] describes a means of sweeping a plane curve B-spline perpendicular to its plane. She then generalizes this method by the introduction of profile curves. Profile curves allow the user to scale the plane curve at different sections along the swept surface. She also gives a detailed account of the necessary exception handling capabilities required, so that the profile curve produces the expected surfaces. Tiller [Til83] in his paper introducing rational splines, generalizes this idea so that a sequence of possibly dissimilar curves can be swept along a spine curve. Woodward [Woo88] generalizes this idea a bit further. He provides a means for the user to modify certain longitudinal properties of the surface once it has been created.

Obviously, once a surface has been created by sweeping or skinning, it can then be modified in whatever way the underlying representation permits. Of course, once modified, the surface must loose its swept characteristics. This is actually quite a drawback, as once again, this leads to a rather one-way design process.

4.1.6 Sophisticated Manipulation of Sets of Control Points

Another interesting line of work has attempted to modify surfaces by means of sophisticated manipulation of the control point net. Barr [Bar84] shows how to modify such a set of control points in order to produce tapers and twists. In fact, in [Smy93] a related method for manipulating a surface is presented along with an implementation. The specific mode of interaction is that the user is able to "press" or "pull" on the surface at a point. By means of a 2D influence distribution in the parameter space as well as the normal at the point, the control points are moved according to their distance from the point of impact.

An arbitrary degree of B-spline local control can be enforced and the "strength" of the push is also, conceptually, a user-definable quantity. Naturally, such a scheme depends heavily on the underlying control point distribution and is is not hard to see that the surface will not be deformed in a position-invariant way depending on where the user chooses to "push". If the point of impact is at a control point, the push will tend to have a greater impact than otherwise.

Sederberg and Parry developed a technique now known as *free-form deformation* [SP86]. Free-form deformation involves placing an object within a control lattice. Moving vertices on the lattice causes the object to be deformed. This can be thought of as a deformation of the space within which the object resides. Coquillart extended this idea so that different types of lattices could be used [Coq90].

Hsu, Hughes and Kaufman [HHK92] developed an interactive system to let the user manipulate one or more control points in which the system solved a least squares minimization in order to determine good positions for nearby points.

Reuding and Sreckovic [RS93] developed a system to allow a user to locally modify a NURBS surface using

- point data
- curves
- surfaces

or any combination thereof. Their idea was to use the least squares method, constrained by the modification input, to produce a smooth result surface.

More recently, Murakami and Nakajima [MN00] have proposed a system in which a user takes advantage of force-feedback technology and, with her hands, directly manipulates a specially created object. Thus, the space of objects that can be modified is very small, but they propose an interesting system nevertheless.

Elsas and Vergeest have developed a system in which a user is able to interactively create displacement features on a preexisting surface [EV98]. Their method includes an interesting scheme for determining the blend surface between the displaced feature and the original surface.

4.1.7 Direct Manipulation of Surface Properties

Tools in this class fall into two categories:

- Modifiers of mathematical properties.
- Modifiers of inspection output.

The first type involves the interactive modification of some mathematical property of the surface directly: such as the curvature, for example. The second type builds on techniques that have evolved to assist in the assessment of the quality of surfaces. For instance, in the automotive design field, reflection lines are frequently used to verify surface quality.

Manipulation of Mathematical Properties

Georgiades and Greenberg presented a system for local modification of surfaces by means of editing the osculating circle radius, and thereby the curvature, at a point [GG92]. In addition they described tools to let a user edit the surface torsion by means of a torsion coil. Georgiades also proposed that surfaces can be edited effectively using a bivariate graph which locally approximates the surface [Geo93].

Manipulation of Surface Inspection Output

A significant amount of work has been devoted to developing tools to let a designer modify a surface by directly modifying the output of a surface inspection technique. For instance, reflection lines are an important verification tool and many researchers have proposed ways of directly displaying these on a model and enabling a user to explicitly edit the shape of the reflection lines themselves, thereby editing the underlying surface appropriately [CBP97, ZC98, LGS99].

Another approach is to generate and modify geometry based upon a user's sketches. Just such and approach was presented by van Overveld [vO96]. He proposed tools that would operate on a surface based upon two-dimensional screen-space sketching. In fact, my research expands on this idea.

4.1.8 Energy Formulations

The fundamental method being employed here is to derive from a conventional parametric surface representation certain characteristics which in turn are optimized in some way. Thus, the surface may be effectively modified internally based on the user's input (in the form of the definition of the objective function). Thus, the user loses direct control over the surface, but, in exchange, gains a modicum of global control over the surface. Conveniently, this gain in control does not come at the expense of a complete loss of local control. In addition, provided the user has a good understanding of the nature of the shape optimization taking place, such a system can be quite flexible.

Thus, where CAGD research has traditionally attempted to build the shape controls into the various representations' basis functionals, this technique effectively relies on a post-processing step based on properties derived from that representation. Thus, although a parametric representation should itself properly be seen as a procedural representation, this is certainly true in the case of representations which seek to optimize surface characteristics. Depending on the implementation and method involved, this post-processing can be quite costly in terms of interaction.

Nevertheless, there are a great many advantages to be found in such a representation scheme. For one, as the objective function being optimized is essentially derived from an existing surface, there is no reason that a system based on such a scheme could not apply different post-processing algorithms depending on the user's need. Further still, there is no reason why the user can not adjust various parameters of the optimization and thus change the "virtual" characteristics of the surface – perhaps, making it more malleable or stiffer.

Many schemes have been introduced which optimize some function based on the curvature over the surface. That is to say, these schemes attempt to distribute the curvature

over the widest possible area on the surface. Celniker [Cel90] correctly observed that there is more to designing "good" surface than merely distributing curvature. He proposes a method of minimizing over the surface the weighted sum of a traditional "bending" term and a "stretching" term related to curvature and area respectively. The area term influences the extent to which the surface will be like a minimal surface while the bending term influences the extent to which curvatures can be distributed over the surface. Thus, he provides a means of user control over the weights in the sum.

Others has have also addressed the user-load problem by introducing an energy minimization step. Ugail, Bloor and Wilson have generalized Bloor and Wilson's blend surface creation work to let a user create and edit surfaces by simply controlling the constraints [BW90b, BW90a, BW94, BW95, UBW99]. Others have also worked along these same lines [MS92, Vas97].

Tension Parameters

Beginning with Schweikert [Sch66], many researchers have pursued the possibility of adding a tension parameter to a spline curve formulation. Schweikert was interested in improving the interpolating capabilities of splines, and, with the addition of this tension parameter, his splines are an analytical solution to the following differential equation

$$\frac{d^4\mathbf{w}}{du^4} - \alpha^2 \frac{d^2\mathbf{w}}{du^2} = 0 \tag{4.1}$$

with given constraints or boundary conditions. Thus, the tension in the scheme is represented by the value of α. It has been observed [Nie74] that this solution is equivalent to finding the $\mathbf{w}$ that minimizes the following integral along the curve

$$E(\mathbf{w}) = \int_C \left(\frac{d^2\mathbf{w}}{du^2}\right)^2 - \alpha^2 \left(\frac{d\mathbf{w}}{du}\right)^2 du. \tag{4.2}$$

Nielson continued and developed a piecewise polynomial interpolant which approximated this $\mathbf{w}$ while interpolating constraints. In both Schweikert's and Nielson's schemes, the tension α for each span can be chosen by the user.

Twist Selection

In transfinite surface representation methods, it is necessary to provide the system with, among other inputs, twist vectors at patch corners. The twist vector is determined by the mixed partial derivative at a corner. Initial schemes set this to zero with the undesirable side-effect of producing *flats*. These are nothing more than clearly visible flatter areas of the surface. A useful compilation of approaches to determining appropriate twist vectors for surfaces was provided by Barnhill, Farin, Fayard and Hagen [BFFH88].

Energy minimization methods have been effectively used to determine better values for this vector. Some efforts have been directed toward determining the optimal twist without changing the position and tangent data already contained in the surface. Hagen and Schultze [HS87] as well as Nowacki and Kallay [NR83, KR90] have published their efforts toward this goal. Hagen and Schultze present a method for smoothing C^2-biquintic Coons patches

which involves determining twist vectors as a convex combination of curvature functions and normal vectors. They ultimately minimize

$$\int_S (k_1^2 + k_2^2) dA \tag{4.3}$$

over the surface. This is his fairness criterion where k_1 and k_2 are simply the principal curvatures. By referring to thin plate deformation theory, Hagen and Schultze observe that equation (4.3) is just a measure of the *strain energy of flexure and torsion* in a thin plate. Thus, minimizing this quantity has the effect of minimizing the strain energy in the surface. They use this fact to derive a formulation based on the calculus of variations technique in order to solve this surface fairing problem.

Others have not restricted their work with the constraint that positional and other tangent information remain fixed. In exchange for this relaxed approach, they have gotten increased surface fairness, but their algorithms have a somewhat more heuristic flavor.

Kjellander [Kje83b, Kje83a] gives both curve and surface versions of an algorithm to improve fairness by moving one data-point at a time according to the value of the objective fairness function. In this particular instance, Kjellander is concerned with Ferguson (Cardinal) splines. He begins by arguing that this spline formulation corresponds to small deflection theory for beams (Bernoulli-Euler Beam Theory). Continuing the physical analogy, he observes that in a physical spline, potential energy is reduced by decreasing the shear forces in it. In the Euler-Bernoulli formulation for beams, the differential equation governing deflections is

$$y^{(IV)}(x) = \frac{q(x)}{EI(x)} \tag{4.4}$$

where $y^{(IV)}(x)$ is the fourth derivative of $y(x)$, the beam's deflection caused by the loading $q(x)$ which may include both distributed and concentrated loads. E is Young's modulus for the beam material and I is the moment of inertia for the beam cross-section. In this case, the potential energy in the beam is only due to strain energy. Thus, as indicated, the Ferguson spline minimizes the internal energy in the same way. Practically, then, decreasing the internal energy at a point by decreasing the shear forces at that point, corresponds to decreasing the difference between the third derivatives on both sides of a point (in the curve formulation). If points on a cubic parametric spline curve are given by

$$\mathbf{r}(t) = (x(t), y(t), z(t)) = \sum_{i=1}^{3} \mathbf{a}_i t^i \; for \; 0 \leq t \leq 1$$

the difference between the third derivatives on both sides of a point is clearly zero if

$$\mathbf{r}_\alpha^{(III)}(1) = \mathbf{r}_\beta^{(III)}(0) \tag{4.5}$$

and this, in turn, is the case for data point i if

$$\mathbf{r}_{ig} = \frac{1}{2}(\mathbf{r}_{i-1} + \mathbf{r}_{i+1}) + \frac{1}{4}(\dot{\mathbf{r}}_{i-1} + \dot{\mathbf{r}}_{i+1}) \tag{4.6}$$

where the g in $\mathbf{r}_{ig}$ denotes the fact that $\mathbf{r}_i$ is a smoothed data point. Moving a data point according to the above equation and readjusting the tangents has the effect of producing a Ferguson curve with smaller strain energy than the original one. Of course, there is a corresponding equation for surfaces but, suffice it to say, that modifying each data point in this way gives an more ideal curve (or surface) in terms of overall fairness. He continues and provides a way of "moving" each point simultaneously so that the problem reduces to solving a set of equations.

Another method proposed by Lott and Pullin [LP88] is similar, but apparently less effective. Again, the same integral as used by Kjellander above is minimized, but with the modification that this is to occur under the constraint that

$$E = \int\int \left[(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2\right]^{1/2} H du dv < \varepsilon \tag{4.7}$$

where H is given by $H = |\mathbf{r}_u \times \mathbf{r}_v|$ (again, $\mathbf{r} = \mathbf{r}(u,v)$ and represents a point on the surface) and where the user specifies a value for ε. This integral is a measure of the volume between the original surface and the new one. In fact, they reformulate this problem in the parameter space of the surface and make a few simplifications. Most important among these is that rather than deal with the full three dimensional problem, they fix one parameter, and vary the others. Thus, they really minimize the potential energy along isoparameter lines. They speculate that the method may work better if applied in full three dimensional space and that further improvements could be gained by adding twist terms although they note that this mixed partial derivative may not be such a good measure of fairness.

Sapidis and Farin [FS88, SF90] describe a method of fairing curves which is something of a hybrid between the schemes of Kjellander and Lott et al. They present an automatic and locally acting preserving smoothing scheme. They set up a global fairness criterion:

> *Fairness Criterion*: A curve is characterized as fair if the corresponding curvature plot is continuous, has the appropriate sign (if the convexity of the curve is prescribed), and is as close as possible to a piecewise monotone function with as few as possible monotone pieces.

Obviously, this criterion is not independent of parameterization and they thus give all of their results in terms of the arc length parameterization. They convert the criterion to mathematical terms as follows:

$$\zeta = \sum_{i=3}^{L+1} z_i \tag{4.8}$$

where $(L+3)$ is the control point count and where

$$z_i = \left|\kappa'(t_i^-) - \kappa'(t_i^+)\right|. \tag{4.9}$$

The curvature is

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{3/2}} \tag{4.10}$$

and from this we can quickly determine $\dot{\kappa}(t)$. Finally, we can get

$$\kappa'(t) = \frac{\dot{\kappa}(t)}{\| \, d\mathbf{s}(t)/dt \, \|}. \tag{4.11}$$

It is clear that this is very similar to the third derivative criterion used by Kjellander. The idea behind their algorithm is to locate the point at which there is the largest κ'-discontinuity and to remove the knot corresponding to that control point in the usual manner of knot-removal [Far90]. Now reinsert the knot to the curve with the knot removed. This procedure produces C^3 continuity (which is equivalent to C^∞ continuity for cubic splines) at the control point rather than C^2.

Gregory [Gre74] devised a scheme in which the twist vector is avoided altogether. Thus, in this representation, singular twist vectors are not a problem. He effectively achieves this through the addition of a rational term which obviates the need for the twist information.

Deformable Models

In general terms, the energy-based techniques referred to above, share the characteristic that they all seek to modify an existing surface. That is, the energy formulation is not embedded in the representation in any real sense. Recently, in the areas which constitute the overlap between CAGD and computer vision, researchers have begun to address the use of energy based techniques in the generation of surfaces. It is perhaps this area of research, into which my research best fits. For an introduction to the field of computer vision, Berthold Klaus Paul Horn's Robot Vision [Hor86] is an excellent source.

In the late eighties Terzopoulos, Witkin and Kass [TPBF87, KWT88] approached the *shape from shading* problem in an unusual, yet very appealing way. They developed a new way of modeling an object with axial characteristics. In turn, this object was made using information from an image which was converted to physical forces which were used to *push* the surface appropriately. The positions of points in a 3D body are given by the time-varying vector-valued function of the material coordinates

$$\mathbf{r}(\mathbf{a}, t) = [r_1(\mathbf{a}, t), r_2(\mathbf{a}, t), r_3(\mathbf{a}, t)] \tag{4.12}$$

$$with \;\; \mathbf{a} = [a_1, a_2, a_3]. \tag{4.13}$$

So, if $\mu(\mathbf{a})$ is the mass density of the body at $\mathbf{a}$, $\gamma(\mathbf{a})$ is the damping density, $\mathbf{f}(\mathbf{r}, t)$ are the net externally applied forces on the body and $\varepsilon(\mathbf{r})$ is a function which measures the net instantaneous potential energy of the elastic deformation of the body, then the Lagrangian formulation of the dynamics of this body is given by

$$\frac{\partial}{\partial t}\left(\mu \frac{\partial \mathbf{r}}{\partial t}\right) + \gamma \frac{\partial \mathbf{r}}{\partial t} + \frac{\partial \varepsilon(\mathbf{r})}{\partial \mathbf{r}} = \mathbf{f}(\mathbf{r}, t). \tag{4.14}$$

On the left hand side of this equation, the first term is the inertial force due to the model's distributed mass, the second term is the damping force due to dissipation and the third term is the elastic force due to the deformation of the body away from its natural shape. This elastic force is expressed as a variational derivative $\partial\varepsilon(\mathbf{r})/\partial\mathbf{r}$ of the potential energy

of deformation which is itself approximated by

$$\varepsilon(\mathbf{r}) = \int_\Omega \sum_{i,j=1}^{2} (\eta_{ij}(g_{ij} - g_{ij}^0)^2 + \xi_{ij}(b_{ij} - b_{ij}^0)^2) da_1 da_2. \tag{4.15}$$

where η_{ij} and ξ_{ij} are weighting functions. The g_{ij} and the b_{ij} are the entries of $\mathbf{G}$ and $\mathbf{B}$, the 2×2 matrices of the first and second fundamental forms respectively. Now, the first variational derivative $\partial\varepsilon(\mathbf{r})/\partial\mathbf{r}$ may be approximated

$$\mathbf{e}(\mathbf{r}) = \sum_{i,j=1}^{2} -\frac{\partial}{\partial a_i}\left(\alpha_{ij}\frac{\partial \mathbf{r}}{\partial a_j}\right) + \frac{\partial^2}{\partial a_i \partial a_j}\left(\beta_{ij}\frac{\partial^2 \mathbf{r}}{\partial a_i \partial a_j}\right) \tag{4.16}$$

where $\alpha_{ij}(\mathbf{a}, \mathbf{r})$ and $\beta_{ij}(\mathbf{a}, \mathbf{r})$ embody the constitutive relations which describe the elastic properties of the material. So that

$$\alpha_{ij}(\mathbf{a}, \mathbf{r}) = \eta_{ij}(\mathbf{a})(g_{ij} - g_{ij}^0) \tag{4.17}$$

$$\beta_{ij}(\mathbf{a}, \mathbf{r}) = \xi_{ij}(\mathbf{a})(b_{ij} - b_{ij}^0). \tag{4.18}$$

When $\alpha_{ij} > 0$, the surface wants to shrink in extent, and when $\alpha_{ij} < 0$ it wants to grow. In this way, the α_{ij} control the surface tensions which minimize the deviation of the surface's actual metric from its natural metric g_{ij}^0. When $\beta_{ij} > 0$, the surface wants to be flatter and when $\beta_{ij} < 0$ it wants to be more curved. In this way, the β_{ij} control the surfaces rigidity which acts to minimize the deviation of the surface's actual curvature from its natural curvature β_{ij}.

They provide us with a sampling of the types of forces to which their model is able to respond. They are able to model body forces (i.e. $\mathbf{f}_{gravity}$ etc.), as well as the connection of points in the body to other points in 3-space by means of Hookean springs ($\mathbf{f}_{spring}$). In addition, they are able to model the effects of viscous forces on the body surface ($\mathbf{f}_{viscous}$) relative to a constant stream velocity and the force due to a collision ($\mathbf{f}_{collision}$). These forces are given by the following expressions

$$\mathbf{f}_{gravity} = \mu(\mathbf{a})\mathbf{g} \tag{4.19}$$

$$\mathbf{f}_{spring} = k(\mathbf{r}_0 - \mathbf{r}(\mathbf{a})) \tag{4.20}$$

$$\mathbf{f}_{viscous} = c(\mathbf{u} \cdot \mathbf{v})\mathbf{n} \tag{4.21}$$

$$where\ \mathbf{v}(\mathbf{u}, t) = \mathbf{u} - \frac{\partial \mathbf{r}(\mathbf{s}, t)}{\partial t} \tag{4.22}$$

$$and\ finally\ \mathbf{f}_{collision} = -\left(\frac{\nabla f(\mathbf{r})}{\epsilon} e^{-\frac{f(\mathbf{r})}{\epsilon}} \cdot \mathbf{n}\right)\mathbf{n}. \tag{4.23}$$

In the equation for $\mathbf{f}_{collision}$, $f(\mathbf{r})$ is the object's inside/outside function and c and ϵ are chosen such that this energy becomes prohibitively large as the model attempts to penetrate itself.

In a subsequent paper [TWK87] the authors apply this idea to a shape-from-shading problem. Based on it, they formulate a symmetry seeking model which they use to extract 3D information contained in 2D images (under certain symmetry-related constraints). In

this way, the surfaces of these deformable models are governed by the variational principle of elastic theory. The fundamentals of the model are as follows:

- A tube of a membrane-like material surrounds a deformable spine.
- Using Hookean spring models, the spine is connected to the tube at regular intervals so that it remains approximately along the "axis" of the tube.
- Next, forces are introduced which *tend* to force the tube into a quasi-symmetric shape around the spine.
- Finally, expansion and compression forces are introduced on the tube which will (in this case) be controlled by image information.

Interestingly, these expansion and compression forces could ultimately be under the user's control which would really allow for the interactive creation of deformable symmetry seeking objects.

The fundamental system has been enhanced considerably by the authors with various other researchers over time [WW94, TQ94, MQV00]. Guillet and Léon employed a variational model to enable the deformation of surfaces using parametric constraints [GL98]. Greiner, Loos and Wesselink developed a variational formulation for surfaces that can be evaluated very quickly [GLW96]. Bloor, Wilson and Hagen investigated the smoothing properties of variational methods when cast as PDEs [BW95].

Blends as Solutions of Differential Equations

An active area of research in CAGD is that of producing *blend* surfaces. A blend surface is a secondary surface which is defined by primary surfaces. So, a blend surface may also be thought of as a transitional surface. While research in this area has taken many directions, one relevant course has been that pursued by Bloor and Wilson [BW89c, BW89a, BW89b]. They have formulated the problem as determining solutions to partial differential equations with given boundary conditions. They perceive that this method could have wide-ranging applicability. Initially, they restricted their work to the area of blend surfaces, however, more recently they have expanded their domain to include surface design in general[BW90b, BW90a, BW94, BW95, UBW99].

Given a finite domain $\mathbf{\Omega}$, with boundary $\partial\mathbf{\Omega}$, they investigate whether a function $\mathbf{x}$ (the blend surface) can be found over that domain such that it satisfies the boundary data. Thus, they are attempting to formulate the problem as the solution to a boundary value problem. In addition, they can impose *behavioral* requirements on the surface (e.g. that it be fair in some sense). As they are looking for smooth solution functions, they concentrate on the class of elliptic partial differential equations. They allude to the fact that numerical solution methods for this class of equations is well established. Thus, they are interested in solving

$$D^2_{u,v}\mathbf{x} \equiv \left(\frac{\partial^2}{\partial u^2} + a^2\frac{\partial^2}{\partial v^2}\right)\mathbf{x}(u,v) = 0 \tag{4.24}$$

subject to the given boundary data, in essence a Dirichlet boundary value problem. With the solution, we have a surface defined in the parameters u and v. The variable a becomes a shape parameter. By varying it, the user is able to eliminate possible *waist* effects in the surface. The solution to the above equation defines a first order blend. Higher order blends are also possible (4th order is given here)

$$D^4_{u,v}\mathbf{x} \equiv \left(\frac{\partial^2}{\partial u^2} + a^2\frac{\partial^2}{\partial v^2}\right)\left(\frac{\partial^2}{\partial u^2} + a^2\frac{\partial^2}{\partial v^2}\right)\mathbf{x}(u,v) = 0 \tag{4.25}$$

which allows us to control additional derivative continuity (in this case, 2nd derivative continuity). A further option is provided by mixed order partial differential elliptic equations.

Finite Elements and a Fairness Criterion

An important area of research has been the application of Finite Element Methods to problems governed by variational principles and partial differential equations.

A very important paper on this subject was published in 1978 by Pramila [Pra78]. His motivation was to improve the design process for ship hulls. He saw that a finite element process could be applied to the problem of surface fairness. He begins by formulating a variational fairness criterion as follows

$$\mathbf{x}(\mathbf{s}) = \frac{1}{2}\int\int \left(w_1\mathbf{s}^2_{,xx} + 2w_2\mathbf{s}^2_{,xz} + w_3\mathbf{s}^2_{,zz}\right) \tag{4.26}$$

where $y = \mathbf{s}(x,z)$ defines the surface of the ship hull (clearly, this particular formulation is specific to the ship building industry, but it could easily be generalized) and the w_i terms are weights. This effectively represents a weighted sum of the squares of the two curvatures in the x and z directions and the twist.

$$k_{xx} = \frac{\mathbf{s}_{,xx}}{(1+\mathbf{s}^2_{,x})^{3/2}} \tag{4.27}$$

$$k_{zz} = \frac{\mathbf{s}_{,zz}}{(1+\mathbf{s}^2_{,z})^{3/2}} \tag{4.28}$$

$$k_{xz} = \frac{\mathbf{s}_{,xz}}{\left[(1+\mathbf{s}^2_{,z})(1+\mathbf{s}^2_{,x})^{1/2}\right]} \tag{4.29}$$

$$k_{zx} = \frac{\mathbf{s}_{,xz}}{\left[(1+\mathbf{s}^2_{,x})(1+\mathbf{s}^2_{,z})^{1/2}\right]}. \tag{4.30}$$

From these, he gets his fairness criterion by arguing that the denominators are all of order one, so that their squares will also be of this order. He justifies this approximation in any event, because the non-simplified measure of fairness is really a subjective metric, so that there is no reason not to make the small modification necessary in order to "linearize" it. Now, with the criterion in this quadratic form, it is possible to manipulate it mathematically

using the Euler equation of $\mathbf{x}(\mathbf{s})$ obtaining

$$\frac{\partial^2}{\partial x^2}\left(w_1\frac{\partial^2 \mathbf{s}}{\partial x^2}\right)+2\frac{\partial^2}{\partial x \partial z}\left(w_2\frac{\partial^2 \mathbf{s}}{\partial x \partial z}\right)+\frac{\partial^2}{\partial z^2}\left(w_3\frac{\partial^2 \mathbf{s}}{\partial z^2}\right)=0 \tag{4.31}$$

and is thus analogous to the partial differential equation for the deflection in a special plate bending problem.

He proposes to use finite element methods to solve the variational problem embodied in the fairness criterion under the prescribed boundary conditions

$$\mathbf{s}(x,z) = \bar{\mathbf{y}} \tag{4.32}$$

$$\textit{and } \frac{\partial}{\partial \mathbf{n}}\mathbf{s}(x,z) = \bar{\mathbf{y}}_{,\mathbf{n}} \textit{ on the surface boundary} \tag{4.33}$$

where $\bar{\mathbf{y}}$ and $\bar{\mathbf{y}}_{,n}$ are given functions and $\partial/\partial\mathbf{n}$ denotes the derivative in the direction of the outward normal at the boundary. Thus, in each finite element we have

$$\tilde{\mathbf{g}}^e = [\mathbf{N}]^e\,(\delta)^e \tag{4.34}$$

where $[\mathbf{N}]^e$ is the shape function matrix for element e (stiffness matrix in typical finite element language) and $(\delta)^e$ is a listing of the nodal degrees of freedom of element e. The shape functions in the above are functions of position, so we can write down the second derivatives as

$$\tilde{\mathbf{g}}^e_{,ij} = [\mathbf{N}_{,ij}]^e\,(\delta)^e \textit{ where } i,j = 1,2,3 \tag{4.35}$$

in indicial notation. Finally, the finite element set of linear equations to be solved is

$$\begin{bmatrix} [\mathbf{K}] & [\mathbf{G}]^T \\ [\mathbf{G}] & [\mathbf{0}] \end{bmatrix}\begin{pmatrix} (\delta) \\ (\lambda) \end{pmatrix} = \begin{pmatrix} (\mathbf{0}) \\ (\mathbf{d}) \end{pmatrix}. \tag{4.36}$$

In this relation, the constraints are written $[\mathbf{G}]\,(\delta) = (\mathbf{d})$ and $[\mathbf{K}]$ is obtained from the next equation. Thus, (δ) and (λ) can be calculated.

$$[\mathbf{K}]^e = \int\int_A w_1\,[\mathbf{N}_{,xx}]^{eT}\,[\mathbf{N}_{,xx}]^e + 2w_2\,[\mathbf{N}_{,xz}]^{eT}\,[\mathbf{N}_{,xz}]^e + w_3\,[\mathbf{N}_{,zz}]^{eT}\,[\mathbf{N}_{,zz}]^e\,dA \tag{4.37}$$

$$[\mathbf{K}] = \sum_{e=1}^{E}[\mathbf{K}]^e. \tag{4.38}$$

In addition to this method, he refers to a penalty method. He mentions certain attractive properties of finite element matrices which he exploits. He observes that the $[\mathbf{K}]$ is symmetric and double banded so that its inversion is computationally cheap. Celniker [Cel90] appears to have hoped that his formulation inherited these qualities. While the matrix Celniker assembles is certainly symmetric and banded, the band-width can become large if the underlying mesh is modified without clever renumbering of the nodes. Thus, as part of this research, a great deal of effort will be given to developing intelligent ways of modifying meshes during user interaction. Thus, in the strictest sense, Celniker's claim is

valid although most practical applications of his idea would necessarily destroy the narrow band-width of the stiffness matrix.

4.1.9 Summary

Clearly, there have been many approaches to mitigating the user load problem. Earlier attempts focused on giving the user better access to the data while more recent efforts have tended to embed fairness criteria into the representation itself. While both of these strategies have their place, it is perhaps surprising that there has been little effort to combine them in a comprehensive way. Thus, allowing a user to work with a surface in the way that best meets the demands of the current stage of the design process. In this research, one of the intentions is to combine some of these methods in a meaningful way, so that a more powerful and interactive surface design system will result.

4.2 Surface Quality

Clearly, the issue of the quality of a surface is pivotal in the field of CAGD. More precisely, the issues of surface appraisal and surface quality are crucial. The user requires not only the ability to create satisfactory surfaces, but also the ability to inspect those surfaces in order to be sure that they are of a sufficiently high quality. There is a great deal that is subjective in this analysis, but it is true that some precise statements can be made concerning what qualifies as "sufficiently high quality". The curvature of a surface is quantifiable, and is usually referred to as the level of smoothness of the surface. The fairness of a surface is more difficult to nail down, but attempts have been made.

4.2.1 Smoothness

The smoothness of a curve or surface is measured by its degree of derivative continuity. Typically, the places on a surface where derivative continuity is most challenged are the joins between patches and at the knot values. This continuity is defined as C^n where is n is the highest integer such that the n-th derivative is everywhere continuous on the surface. Strictly, C^1 continuity implies C^0 continuity, but if we have C^1, C^0 is not so impressive. Thus, we say that a curve with C^n continuity is *smoother* than one with C^m continuity where $m < n$.

The level of continuity required depends on the application of the surface. For example, objects to be machined require at least C^0 continuity but generally have higher levels. An automobile surface is designed so that the perceived reflection lines on the surface are themselves C^1 continuous. This requires C^2 continuity of the surface itself as the reflection lines are influenced by the surface normal which is itself based on the first derivative of the surface. Kaufmann and Klass, and Poeschl [KK87, Poe84] provide interesting accounts of the use of reflection lines in the automotive design process. Poeschl explains the usefulness of reflection lines in indicating surface quality while Kaufmann and Klass take this a step further and describe a system in which the user is able to interactively edit these isophotes and thereby modify the surface normal at a point and thereby the surface shape in that region. They also addresses the problem of the oscillations that can arise in their system.

Aerodynamic surfaces also require at least C^2 continuity if not higher levels. Finally, kinematic surfaces such as those of cams require at least C^3 continuity as not only acceleration terms must be continuous, but also *jerk* terms. There is an excellent discussion of such surfaces and their design provided by MacCarthy [Mac88].

When considering parametric surfaces, determining true derivative continuity is not a trivial matter, nor is enforcing strict derivative continuity. As discussed earlier, the reason is that a single surface can have many parameterizations and in general, the derivatives will not be the same at corresponding points. Barsky and Beatty [BB83] coined the term geometric continuity (sometimes called visual continuity) as a response to this continuity problem. While the derivatives of identically shaped surfaces with different parameterizations are in general not the same, the directions of the derivatives at corresponding points are, in general, collinear, although not necessarily with the same sense. Thus, Barsky and Beatty convincingly argue it makes sense to talk about geometric continuity. A C^1 continuous curve is said to be G^1 continuous, for example, but the reverse is not generally the case. In general, a G^1 curve is **not** C^1. Thus G^n continuity is concerned with unit n-th derivative vectors. Many have argued that it is the geometric continuity and not the parametric continuity that is important in determining the quality of a surface. This is very convenient, as geometric continuity is much easier to determine. In fact, Bézier curves have the property that one can write down the geometric implications of C^n continuity directly [BFK84]. Cubic B-splines have a similar property.

Farin [Far82a] gave the necessary algorithms in order to enforce geometric continuity in curves and surfaces. This theory was extended by DeRose [DeR85] and Lee and Ravani [Lee90].

4.2.2 Fairness

As suggested above, while smoothness has a "value", fairness is a subjective appraisal of a surface. Thus, many have attempted to formulate mathematical conditions which capture fairness, but it remains a subjective term. Many of the techniques which subjectively fair a surface have been mentioned already.

4.2.3 Visual Fairing

If fairness is a subjective quantity, then it is natural to consider interactive fairing of surfaces. Thus, research has been done to give the user information which is valuable in this task. The aim is, assuming sufficient information is given to the user, the control points defining the surface can be modified until a satisfactory level of fairness is reached.

On-Screen Inspection

Beyond the usefulness of standard rendering tools, such as the Blue Moon Rendering Tools (BMRT) [Gri00] in the visual evaluation of surface quality, Beck, Farouki and Hinds propose that the user needs to be alerted to particular surface features using various surface display techniques [BFH86]. Beck et al. indicate that merely displaying isoparametric curves on the surface is inadequate toward this end. They suggest the use of

- patch boundary display,

- surface contouring,
- shading,
- differential geometry information,
  - curvature maps (Gauss, principal, mean)
  - lines of curvature
  - geodesics (rather complex)
- as well as isoparametric lines

in order to analyze a surface in terms of its fairness. Munchmeyer confirms that these are very useful tools in his study of surface imperfections [Mun87]. Bonfiglioli extends this palette of tools with the addition of the observation that the silhouette of an object is often very revealing [Bon86]. An algorithm for determining the silhouette of an object in a given view is provided although from the point of view of surface analysis, it is worth noting that the algorithm is very heuristic in the sense that it only gives an approximate silhouette based on the control net of the surface. Nevertheless, it is a rather appealing algorithm in its own right.

Various researchers have explored the physiological issues relating to the perception of form. Stevens investigated the ways in which we understand surface contours [Ste81]. Pentland explored more general questions of how we perceive form in [Pen86].

As already mentioned above, Poeschl [Poe84] refers to the usefulness of isophotes in analyzing a surface while Kaufmann and Klass [KK87] confirm this conclusion and describe a system by which the user can interactively modify the isophotes themselves and, in this way, modify the actual surface. Hutchinson and Hewitt proposed a very clever optimization of this basic technique by using false color and hardware displays [HH96]. Lennings, Peters and Vergeest created a unified approach to this class of display [LPV95].

Others have determined that it is useful to display actual mathematical properties of surfaces and then use this feedback to make further modifications. For instance, Theisel and Farin [TF97] propose the use of all of the following

- contour lines,
- curvature,
- geodesic curvature,
- asymptotic lines, and
- isophotes

to aid in assessing the quality of surfaces. On a related note, Maekawa, Wolter and Patrikalakis propose the use of umbilics in conjunction with lines of curvature as further interrogation aids [MWP96].

Physical Inspection

Another very effective way to inspect surface quality is to physically produce the object being inspected. Numerical Control (NC) milling was one of the original motivations behind the efforts of Bézier and de Casteljau to come up with a computational model for freeform surfaces. Liu provides techniques that are well-suited to to the milling of freeform surfaces [Liu95].

Convexity Based Fairing

Another interesting idea is presented by Bedi and Vickers [BV89] and, once again, they are interested in the domain of ship hull design. In their system, a hull is represented as a skeleton of what are basically section curves (in both planes). Rather than simply using these to generate a surface, they suggest analyzing each of the skeletal curves individually in terms of the divided differences of the nodes on the curve. They propose an interactive system. Given a curve $y = f(x)$, the first and second divided forward difference are, respectively,

$$g(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \; and \tag{4.39}$$

$$h(x_i) = \frac{g(x_{i+1}) - g(x_i)}{x_{i+1} - x_i}. \tag{4.40}$$

By inspection, they look for the node with the largest deviation and move the node to compensate. This scheme bears some resemblance to the energy based smoothing algorithms described in section 2.1.6, but suffers from having smaller support and not being automated in any way.

A number of researchers have postulated that fairness problems in curves are related to spurious points of inflection in them. While it may not have been satisfactorily demonstrated that this is indeed a valid postulate, the techniques employed in this regard are very interesting and are therefore worth noting.

Perhaps most novel among these techniques are those developed by Hoschek [Hos84, Hos85]. He is interested in detecting regions of zero-curvature on the surface, and thus, enforcing convexity. In this review, I will give the 2D version of the algorithm although he does provide a 3D generalization of it. Essentially, the technique involves viewing the curve in a type of dual space. He is interested in the *polar image* P of a curve. Polarity maps share with duality maps the property that a point is mapped onto a line and vice-versa. He uses a mapping defined with respect to the unit complex circle. Thus, the image of a point $\mathbf{x}$ at parameter value $t = t_0$ on curve $\mathbf{x}(t) = (x(t), y(t))$ under this polarity map is the (polar) line:

$$\xi x(t_0) + \eta y(t_0) + 1 = 0. \tag{4.41}$$

By differentiation (when t runs through its range)

$$\xi = \frac{-\dot{y}}{x\dot{y} - \dot{x}y} \; and \; \eta = \frac{\dot{x}}{x\dot{y} - \dot{x}y} \tag{4.42}$$

The way of choosing ξ and η (as well as the ζ necessary for surfaces). If x is a distance $d = \overline{OX}$ from the origin, then the polar line corresponding to x is a distance $d_p = \frac{1}{\overline{OX}}$ from the polar origin. Then the 2D theorem upon which Hoschek's technique is based is

Theorem 2D 1 *If the plane curve* $\mathbf{x}(t)$ *has an inflection point at* $t = t_0$*, the polar curve* $\mathbf{P}(t)$ *has a singularity (cusp) at the corresponding point.*

The 3D theorem is a rather straight-forward generalization of this theorem. One drawback of this method is that it relies on user specified correction. In this sense this method might well also belong in the Visual Fairing section.

Hoschek's second paper on this subject [Hos85] presents a much more efficient way of detecting zero curvature than with the polar map of the curve using a complex unit circle. He suggests using the *k-orthotomic* mapping of a curve (surface). In his words:

> We assume that a plane curve $\mathbf{x}(t)$ is given and a $\mathbf{P}$ which is not a point of $\mathbf{x}(t)$. Now we reflect $\mathbf{P}$ in the tangent line of $\mathbf{x}(t)$ at a point $\mathbf{x}(t_0)$ and get the point $\mathbf{y}_2$. Hence, as $\mathbf{x}(t)$ varies, the point $\mathbf{y}_2(t)$ moves along a curve: this curve is called the orthotomic of $\mathbf{x}(t)$ relative to $\mathbf{P}$. We can interpret the orthotomic also as the reflected wavefront of light emanating from $\mathbf{P}$ or the locus of apparent positions of $\mathbf{P}$ viewed in the mirror $\mathbf{x}(t)$.

Thus, in general, the k-orthotomic is given by

$$\mathbf{y}_2(t) = \mathbf{P} + k\,[(\mathbf{x}(t) - \mathbf{P}) \cdot \mathbf{N}(t)]\,\mathbf{N}(t). \tag{4.43}$$

A similar theorem holds for this method as did for the polar map scheme. The curve $\mathbf{y}_2(t)$ shows a cusp if the curvature of $\mathbf{x}(t)$ vanishes as before. Again, this scheme has a surface generalization. (Chang [CF84] proposed a convexity criterion over triangles).

As mentioned earlier, Andersson et. al. [AAB+87] provide an automated algorithm for converting a general surface to a convex one. The algorithm seeks to make the Gaussian curvature be of the same sign everywhere on the surface by moving control points normal to the surface within sufficiently small tolerances. In this work, the problem is cast as a linear programming problem and Karmarkar's algorithm is applied in order to solve this system.

Another, albeit less innovative approach was presented by Ferguson, Frank and Jones [FFJ88] in which they proposed a formulation for a transformation from a good surface to a convex one. This formulation is achieved through a non-linear optimization. They examine projections of isoparameter curves in the surface onto user supplied planes. Each of these projections is made convex with the constraint that the sum of the squares of the distances from the old curve to the new curve is minimized. By performing this on some selection of iso-curves, the surface is transformed and guaranteed to be convex in the corresponding directions. Clearly, the good choice of projection planes for the isoparameter lines is very important and perhaps, this method is not quite as automatic as it should be.

Energy Based Formulations

Many of the formulations described in the previous section were developed as a means of addressing the surface fairness problem. Thus, many of the energy formulations involve

terms which depend on the curvature of a surface which in turn provides a measure of the internal potential energy of the surface.

4.3 Triangular Elements and Their Interpolation

While many researchers have long advocated the use of triangular patches as the underlying structure for surfaces, the fact that rectangular tensor-product surfaces are so well-understood and relatively simple to implement has made it difficult for triangular patches to become accepted. Perhaps, rectangular schemes are so popular in part because they are such a direct extension of the standard curve scheme. However, when compared with triangular patches, rectangular patches suffer from the severe handicap that they can model only a smaller set of surface patches. This is very important in many practical applications. Systems which rely on rectangular patches have great difficulty representing pentagonal patches, and even if they are able to, it is usually done via a complex "special case" involving singularity functions. Clearly, it is desirable to avoid such special cases and the notion of having a "canonical" surface which can represent all surfaces that the system can handle is very appealing. The triangular patch is just such a canonical representation, and it is thus preferable. Obviously, the shape function of the element is important in any finite element formulation, but this section will concentrate on the triangular element as a surface element.

4.3.1 Approximation Schemes for Irregular Meshes

It was with a triangular patch that de Casteljau originally extended his Bézier scheme to surfaces. Catmull and Clark [CC78] subsequently attempted to set down an improved shading technique based on a recursive algorithm for producing a B-spline surface approximating an arbitrary topological grid. Catmull had already developed a similar scheme for topologically rectangular grids but he realized the need for more general algorithms. The fundamental idea is a fairly straight-forward geometrical construction. The single caveat is that although the surface is almost everywhere C^2 continuous, they can not guarantee this at a few isolated *face* points. Nevertheless, they were willing to speculate that, based on the shaded images they created, it appeared as though the surface was at least C^1 at these points. However, they offered no proof of even that limited claim.

Jörg Peters has proposed a number of non-interpolating subdivision schemes that produce surfaces over irregular meshes [Pet95, Pet96]. Loop also proposed a popular approximating subdivision scheme [Loo94]. While there are many other, subdivision has become a popular surface representation in application domains other than merely data approximation. For instance, they have found wide-spread applications in the creation of special effects due to the fact that their use obviates the need for surface stitching.

Navau and Garcia have recently developed a method for creating B-Spline surfaces corresponding to irregular control meshes that can be embedded in a plane (in other words, *manifold* meshes) [NG00].

4.3.2 Triangular Patches in General

Farin has long been an advocate of the triangular patch for many of the reasons given above and in [Far82b] provides a nice introduction on the subject of triangular Bernstein-Bézier patches. By analyzing the cubic Bernstein-Bézier patch with barycentric coordinates u_1, u_2 and u_3,

$$\mathbf{P}(u_1,u_2,u_3) = \sum_{\substack{i+j+k=1 \\ i,j,k \geq 0}} \frac{3!}{i!j!k!} u_1^i u_2^j u_3^k V_{i,j,k}^3(u_1,u_2,u_3) \tag{4.44}$$

$$where\ 1 = u_1+u_2+u_3 \tag{4.45}$$

he defines the conditions which will guarantee C^1 continuity between such patches. In [Far86] Farin provides a thorough survey on the history and algorithms which relates to triangular Bernstein-Bézier patches in CAGD. He discusses the different interpolants which have been developed for triangles. He refers the the Clough-Tocher [CT65] C^1 interpolant and the necessary split for the triangle. He suggests an improvement on this element which would make it very similar to Celniker's 12 degree-of-freedom (DOF) triangular element [Cel90]. He discusses the Powell-Sabin interpolants. He provides generalizations to n-dimensions (simplex) and discusses parametric variations on many of the methods. It is a very rich source of information on triangular interpolants.

In an excellent paper on the subject of higher degrees of continuity than C^1, Whelan [Whe86] provides an extremely detailed and useful description of a C^2 triangular interpolant. This work contains a very precise description of how to derive the coefficients of the 55-DOF element. She gives a no-frills 9-th degree version of the full system. However, she also provides an excellent account of the process by which she statically condenses this interpolant to a 7-th degree polynomial. She is again considering the Bernstein-Bézier triangle.

She begins by remarking on the quintic interpolant, that it only provides C^1 continuity. This is the basis of the 18-DOF element and has been extended to a 21-DOF element by the addition of the normal derivative at the midpoint of the edges to the position, 1st and 2nd derivatives at the vertices. Thus, it is reasonable to look for a C^2 interpolant over the triangle. She refers to Ženišek's proof that the simplest C^2 interpolant over a triangle is of degree 9 [Žen72]. Ženišek was the first to provide general theorems regarding C^n triangular finite elements. Based on the barycentric description of the Bernstein-Bézier interpolant over triangles, Whelan provides formulae for the derivatives within the triangle in arbitrary directions. The condensed scheme described in her paper requires only vertex information up through the 4-th derivatives.

Gregory and Charrot [GC80] developed a C^1 triangular interpolant related to the Coons patch idea. The underlying idea here is to blend three triangular patches. Each of the triangular patches is determined by the tangency conditions along two of the edges of the triangle. They apply it to the area of fillet creation, however, and it remains rather messy. Gregory [Gre85] generalized this C^1 interpolant for a simplex of any degree.

Jones [Jon88] shows how to decompose patches with any number of bounding edges into a set of rectangular patches. Storry and Ball [SB89] make a similar effort, but here

B-splines are used instead of blending functions. Zheng and Ball developed a method for producing multiple four-sided Bézier-like patches corresponding to non-four-sided surface patches [ZB97]. Bloor and Wilson [BW89c] define a surface as the solution to a partial differential equation and developed a convolution mapping in order to turn n-sided holes into squares. Then, for the final solution, they apply an inverse convolution to the square. Based on a forward differences scheme, this method produces a surface represented by a set of discrete points. Herron [Her85] observed that requiring C^1 continuity between patches only makes sense if the patches are, in fact, adjoining. He proposed a G^1 triangular interpolant in order to overcome this shortcoming. Another technique for using normal tensor product patches to represent non-rectangular patches has been that of using *trimmed* surfaces. A trimmed surface is nothing more than a regular tensor-product patch restricted to an arbitrary set of closed areas of the parametric domain. This domain can be of any shape. Casale [Cas87] described such a scheme.

4.3.3 Triangular Finite Element Interpolants

Many triangular interpolants were originally developed in the field of Finite Element Analysis. The flexibility of the triangle made it an early favorite of researchers in that field. While current finite element systems frequently also allow the capability of having rectangular (and other) elements, it is worth noting the development of triangular interpolants in this field.

Clough and Tocher [CT65] provide a very useful survey of elements which were in common use in the mid-sixties. They review 3 rectangular elements and 3 triangular elements. In addition they present a new element which they also compare with the existing ones. The rectangular elements under consideration are the Adini and Clough element (ACM), the Melosh element (M) and the Papenfuss element (P). The 3 triangular elements being reviewed are the Adini element (A), the Tocher element (T), the Tocher-10 element (T-10) and finally their new element which they call Hsieh, Clough and Tocher element (HCT). According to the authors, Hsieh had the original idea for this element, but it was developed by Clough and Tocher. In subsequent work, authors often refer to this element as simply the Clough-Tocher element. The basic idea behind the 9 degree-of-freedom element being introduced here is that the triangle is subdivided into three smaller triangles. This element is related to the 9-DOF element which Celniker [Cel90] employed. They lay out some basic guides to be followed in the selection of compatible element deformation functions (p516). They briefly describe a method of determining the stiffness matrix for this element. There is an interesting undercurrent discussion of the twist effect in the fully expanded polynomial (in x and y) expression which corresponds to the xy term. In this new element, there is no xy term in each of the displacement expressions for each of the three sub-elements. It is for this reason that the slope may only vary linearly across the exterior boundary of the triangle. Finally, it is also for this reason that they can guarantee slope-compatibility in a system of such elements.

Each sub-triangle effectively has 9 degrees-of-freedom and thus the overall triangle would seem to have 27. Clough and Tocher provide a detailed argument for why this apparently 27-DOF system is really only a 9-DOF system. It seems as though this is an early variety of static condensation of the interior degrees-of-freedom. The tests that the author performs

on the various elements mentioned above, lead them to conclude that their triangular element behaves almost as well as the then extant rectangular elements. In addition, theirs has the obvious advantage of being more readily capable of representing a larger set of surfaces. Additionally, the triangular element converges in a strictly monotonic fashion toward the correct solution, albeit slightly more slowly than the rectangular elements which do not converge monotonically to that solution. Finally, the authors speculate on a 12-DOF triangular element which is formulated in a similar manner to their 9-DOF triangle:

> It is of interest to note that further improvement is possible in the compatible triangle if one is willing to employ additional nodal points in defining its degrees of freedom. For example, a 12-DOF triangular element has been developed in which the three additional degrees of freedom are represented as the normal slopes at the midpoints of the sides. In this case, a complete 10-term polynomial expression may be assumed for the displacement functions of the sub-element, and the normal slopes along the boundaries of the element may vary quadratically rather than linearly.

It is clearly a very similar logic that Celniker followed in developing his 12-DOF triangular element.

The Clough-Tocher element has become so ubiquitous that many researchers have been improving on it. Farin modifies it by reformulating it as a Bernstein-Bézier patch [Far85]. Kashyap developed an 18-DOF Clough-Tocher element with improved smoothness properties [Kas96]. Mann, in turn developed a C^2 version of the Clough-Tocher element [Man99].

Bazeley et al. [BCIZ65] develop a somewhat more sophisticated triangular element also with 9 degrees of freedom. Their element is developed for thin plate bending problems and provides C^1 continuity across element boundaries. At a more theoretical level, they question the conditions set down in [CT65] in order to guarantee convergence. They correctly claim that the previous condition was too restrictive and that many of the existing rectangular elements did not satisfy it – yet many of them do converge to a solution. In an attempt to explain this inconsistency, Bazeley et al. propose a new condition sufficient to guarantee convergence:

> The new condition is that of the displacement function being capable of representing constant curvature (strain) states throughout the finite element irrespective of its size or shape.

Many call this condition the *completeness* condition and some "add" to it the requirement that the interpolation be able to represent zero curvature (zero strain) states throughout the finite element [Bat82, page 167]. However, a zero curvature state is obviously also a constant curvature state. Bazeley et al. also describe an extension of the 9 degree of freedom element to a 12-DOF element in a very similar way to that described by Clough and Tocher.

While C^1 continuity in triangular elements may be enough in itself for many applications, it is not enough to ensure that a surface represented will have a pleasing appearance. In fact, Celniker [Cel90] observed unsatisfactory flat areas in surfaces constructed using the 9-DOF interpolant he described. This aesthetic problem was redressed in Celniker's thesis by development of the 12-DOF element which, although it remains C^1, produced a

more pleasing surface. Whether or not this is a sufficient compensation for a loss of C^2 continuity is not clear. Effectively, Celniker has attempted to get C^2 continuity through the constitutive relations he defines for the surface. It is a very clever idea, but it needs to be tested.

Irons and Bell independently developed 18-DOF C^1 interpolant with quartic deviations in edge shape and cubic variation in tangent normal to the edge direction. The two interpolants differ in the choice of degrees of freedom. Liu and Zhu has provided an overview of C^2 triangular interpolants [LZ95].

Chapter 5

The Implementation

This chapter covers two important aspects of the implementation. First, it describes the underlying surface representation used in the implementation, along with the reasons this particular representation was selected. Second, it details the new tools developed, both from the perspective of the designer and that of a software developer.

5.1 Underlying Surface Representation

The surface representation employed here is a finite element system built upon an irregular triangulated mesh. There are a number of good reasons for this choice, but it must, nevertheless be understood that this choice was in no way inevitable. There is no reason the surfaces could not have been implemented with an interpolatory subdivision scheme (on irregular meshes) such as the Modified Butterfly Scheme introduced by Dyn, Gregory and Levin [DLG90], with hierarchical B-splines or even with triangular B-spline patches. The finite element approach offers certain advantages over each of these but it may also carry disadvantages when compared to them as well. Suffice it to say that a system very similar to the one implemented here ought to be possible given a choice of some other interpolatory representation scheme as long as its surfaces can be defined in terms of an irregular mesh.

5.1.1 Features Required of Representation

An useful way of understanding my choice of representation is to list the important features that the representation needs to support. Although all of these features are not fully exploited in this implementation, they are nevertheless desirable and influenced my choice.

Representation need not be defined on a regular grid

Most B-spline representations in use today are defined over a regular grid. Thus, if a designer needs to add detail within what is currently a coarse large 4×4 surface such as that in Figure 5-1, she must split the surface into nine smaller surfaces, each of which has 4×4 control points, as shown in Figure 5-2, in order to have a sufficient number of control points in the area of interest. Thus, whereas the designer only needed 12 additional control points in the surface, she now must continue her work with an additional 84 distinct

control points and 12 simultaneous edge-edge stitching problems. Clearly, the need for this additional detail has weighed down the model significantly with little real gain.

It is important to note that the nine surfaces created in Figure 5-2 are a "best-case" situation in which a simple 4×4 control net (for a cubic B-spline surface) will be sufficient to capture whatever detail is required. For instance, it might have been the case that in order to continue to develop a particular design with a sufficient degree of control, the designer might well have required far more than a 4×4 control net in the central surface. Had that been the case, the increased density would have been propagated to the four sliver surfaces that each share an edge with this new central surface.

As if this were not bad enough, there is an additional cost, for these nine surface now must be stitched after almost any edit if any degree of continuity is required between the patches.

It should be noted that while other techniques have been developed to address this problem, these either involve more difficult cross-edge constraints (when the number of control points along a shared edges differs in the two surfaces sharing the edge) or the use of trimmed surfaces which introduce their own host of problems when continuity is needed across surface-surface boundaries.

In this implementation, the ability to add detail in sub-regions of surfaces was extremely important from the very beginning and it is for this reason that the surfaces are represented over an irregular mesh. While I probably could have used the modified butterfly subdivision scheme [DLG90] to address these concerns in my implementation, I chose to address them using the finite element scheme proposed by Celniker [Cel90].

Avoiding the need for stitching

Spline surfaces of all types suffer from the drawback that in all but the simplest cases, adjacent surfaces require complex stitching operations so that the surface they are representing retains a degree of continuity across patch boundaries. Thus, creating features that straddle patch boundaries is extremely difficult – both for the user and for the programmer.

This is an amazingly important issue and one which has been avoided to a great degree in the literature and even in commercial applications. Avoided not in the sense that researchers have not developed effective stitching schemes, but in the sense that little has been done to get away from the need for stitching altogether. Yet, the recent interest in subdivision surfaces is finally beginning to address this problem.

In fact, the reality is that if a change is required that would cross an edge, it is often much easier to simply delete the surfaces and begin again while making sure to configure the patches so that the new feature will completely fit in one patch.

This is clearly quite an impediment to a fluid design process and this is, in fact, a reason many industrial designers will not use computational tools until the essence of the design has been resolved. In other words, they are opting not to use the computer in the design development process, but rather in the design presentation process.

Support Level-of-Detail

Another useful feature in the representation would be the ability to support different levels of detail in some natural way. In making this observation, however, I am not referring to

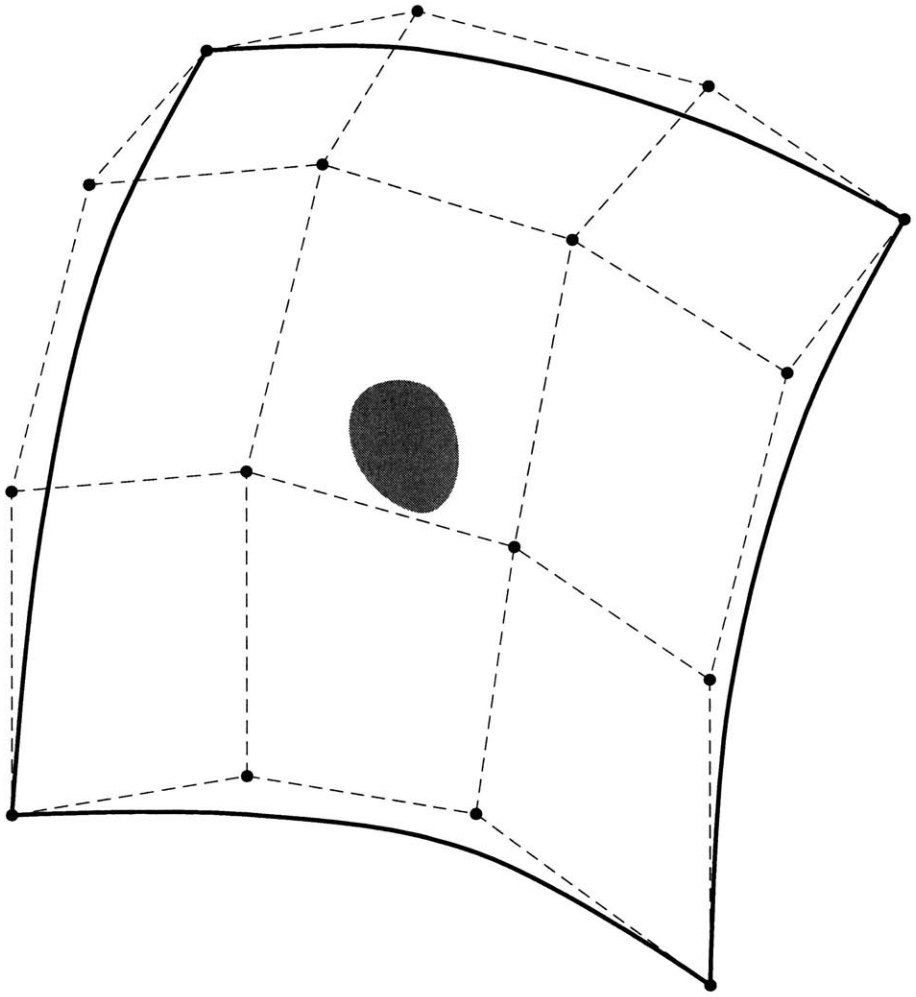


Figure 5-1: B-spline surface with region in which more control is required.

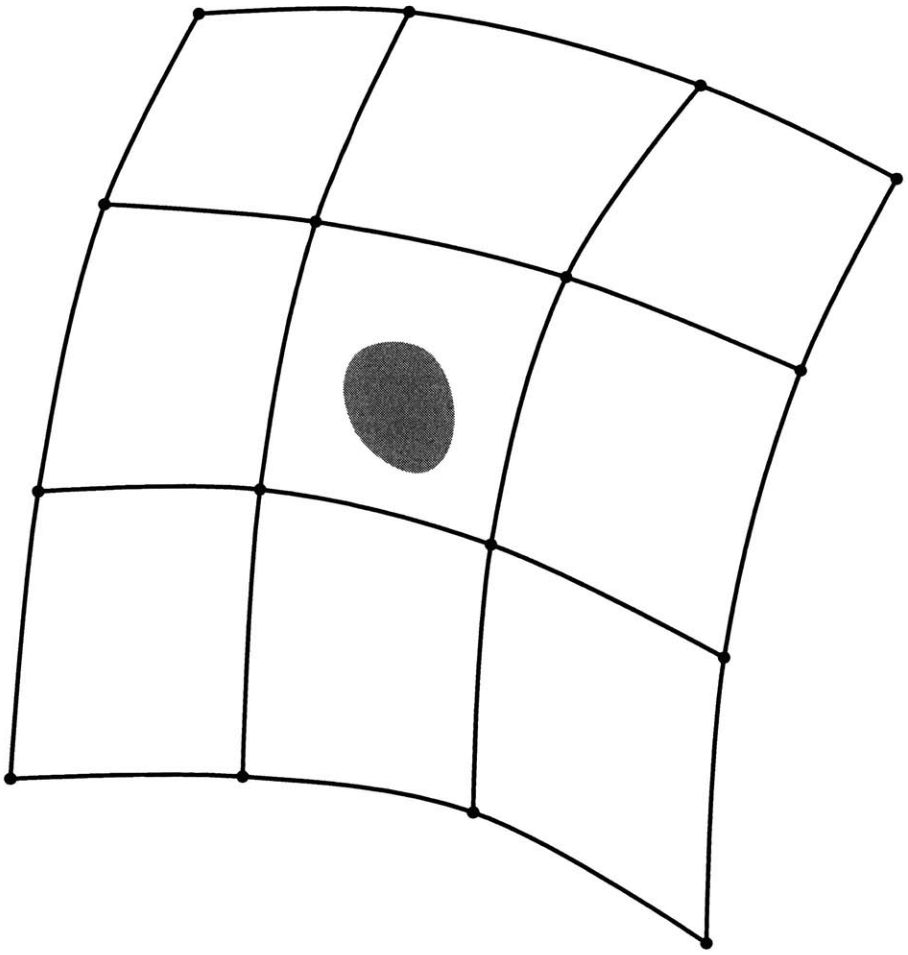


Figure 5-2: Nine B-spline surfaces created in order to provide designer with increased control in a region

level-of-detail in the traditional computer graphics sense. Traditionally, level of detail has come to describe the process of changing the complexity of the model in order to minimize the number of triangles rendered without compromising the quality of the rendering. This type of display-performance level-of-detail has been an area of research within the computer graphics community.

While this may be an interesting feature in the implementation as well, it would be far more useful to have a representation which is as interactive as possible in the early stages of design – when the designer needs to rapidly explore wildly varying ideas with a minimum of impediments. In effect, the user would add detail in the representation as the design develops.

Direct access to surface properties

Many current surface representations such as B-splines and the most popular subdivision schemes do not have surface points and derivatives as direct degrees of freedom. Thus, editing such surfaces must always be done indirectly: the user pushes and pull controls points in order to modify the surface shape. Of course, researchers have expended significant effort in order to let the user do seemingly natural things like pick a point on a surface and move it so while the underlying surface is modified subject to the constraint that it pass through that point. Yet, such constraint-based approaches are really ruses since the user is only indirectly making those edits while the system is generally solving a system of equations. Hence, for the purpose of enabling such "direct" manipulation, the surface is temporarily imbued with physical properties only to lose them as soon as the edit is complete.

5.1.2 Additional Features Desired in Representation

A number of other factors were taken into account in choosing the finite element surface representation for this implementation.

Interactive Speed

The speed of the underlying representation was certainly an important issue. Yet, it has become clear to me that issues of interactive speed as well as local control have become somewhat muddied. That is to say, although interactive speeds are always expected of applications and new techniques, this expectation is not very meaningful without properly describing the amount of work that can be completed at an interactive speed. Thus, while it is true, for example, that a B-spline surface can be updated very rapidly when moving control points, it is also true that in a system of patches, there is a great deal more that needs to be done in order to realize a smooth surface using a B-spline representation. For instance, stitching may still be needed, not to mention the amount of work that may be needed to meaningfully move a control point: for example, the surface may need to be split into nine smaller surfaces, transforming what should have been a simple control point move into a 10 control point move in conjunction with a 9 patch stitch operation. Very few systems could handle the general case of such a problem at interactive speeds.

On an SGI O2, reasonably complex surfaces represented by the finite element system employed in this system are not truly interactive (multiple frames per second) while dragging

a point on the surface for a reasonably complex surface. However, they are fast enough to be called very near interactive. And, in any event, in light of the discussion above, they may well be significantly faster than a surface represented by a B-spline.

Smoothness of the surfaces

The 12-DOF triangular interpolant used in this implementation guarantees C^1 continuity across the triangle boundaries. However, as Celniker points out, the minimization of the integral over the surface tends to distribute curvature over large areas and, therefore, the surfaces are likely to seem smoother than a C^1 surface ordinarily would (see page 19 in [Cel90]).

There exist triangular elements with higher degree-of-freedom counts that deliver true C^2 surfaces. Implementing at least one of these would be an obvious future project, although it has not proven to be important in this work up to this point.

However, similar to the comments made above with respect to interactivity, surface smoothness may also be something of a red herring – at least where industrial design is concerned. From the designer's perspective, C^2 continuity is, in itself, not terribly important. All that matters is that the surfaces look and feel good when judged against whatever standards the designer chooses. For instance, a highlight line inspection as well as running one's hand over the surface are common tests.

It would certainly be desirable to support such inspection.

5.2 Primary Tools in the Implementation

The software created during this research is primarily intended to demonstrate the basic validity of the three-dimensional tools discussed in detail in Chapter 2 (see 2.5.2 on page 63).

Thus, while developing a Sketch Tool, a Tape Tool, and a Tape-To-Model Tool would all be interesting explorations in themselves, this research has been confined to demonstrate the usefulness of the following tools:

- Section Sweep Tool
- Dragging Sweep Tool
- Shape-from-Shading Tool

In addition, the following tools have been implemented[1] in order to enable better use of the three important tools listed above:

[1]In fact, I also implemented a Sketch Projection Tool for BMW AG., Munich, although this is not formally part of this research. The tool I developed there was written in about a month and was intended as a proof-of-concept. BMW has since done considerable development work on the tool and it has become an actively used tool in their design process.

The concept is simple. Given a rough model and one or more sketches as input, provide the designer with interactive tools to control the position and properties of a virtual slide projector associated with each sketch. Thus, using the interactive tools, the user can project each of the sketches onto the rough model. Once the sketches (textures) have been properly placed, they are locked down and the textured model can now be viewed in any suitable environment.

- Define Region Tool
- Create and Embed Curve-On-Surface Tool
- Move Point-On-Surface Tool
- Move Point-On-Surface Along Normal Tool.

5.2.1 Section Sweep Tool

This tool is a natural extension of sweep tools available in many commercial systems today. It is not at all uncommon for a B-spline based surface modeling system to provide users with a means of generating surfaces by letting them create cross-section curves. These cross-sections will generally either have a precise ordering or they will be placed along a spine curve. Piegl and Tiller describe these and related techniques in considerable detail in [PT97]. Curve network interpolation schemes can be viewed as a generalization of skinned surfacing.

However, in the case of all of these advanced surface creation techniques, their purpose is just that: surface creation. In the course of researching freeform surface design methods, it very quickly became clear that what is sorely lacking in current systems is not a greater number of surface creation techniques, but rather a more sophisticated set of surface modification tools.

The Section Sweep Tool addresses this shortfall by extending the idea of lofting to a surface modification tool. Before using this tool, the user defines a curve on the surface to be deformed. The system allows the user to do this by placing points on the surface so that as soon as at least three of them have been defined, a curve is generated that interpolates the parametric locations in the domain. The corresponding three-dimensional curve-on-surface is drawn as shown in Figure 5-3.

The user enters the tool and places *sails* or section planes along the curve-on-surface at any desired locations. The only requirement is that there must be at least two sections. In this way, the system has a natural spine curve as well as an ordering of the sails. As each sail is placed, it becomes the currently active sail and the control points for the curve defined within that section are modifiable. Also, within a smaller window (which can be called up as an option), the view is automatically changed so that it is "looking" at that section (see Figure 5-4). In an active section, control points can only be moved within the plane defined by the section plane.

The plane for a given cross-section is derived from the three-dimensional tangent to the curve-on-surface at the given point on the curve-on-surface corresponding to the sail. The up-vector for the plane is defined to be the normal to the surface at the given point. And finally the third basis direction for the plane is the cross-product of the two geometrically determined vectors.

The user has the ability to toggle between already defined sections by simply selecting them, or, more practically, using the Up-Arrow and Down-Arrow keys to sequentially select them. Once the user is satisfied with the cross-sections defined in each sail, she may trigger the modification. In this case, the underlying mesh density was insufficient to properly capture the detail implied by the defined cross-section. It can be seen in Figure 5-5 that the surface has only been modified in the most minimal way.

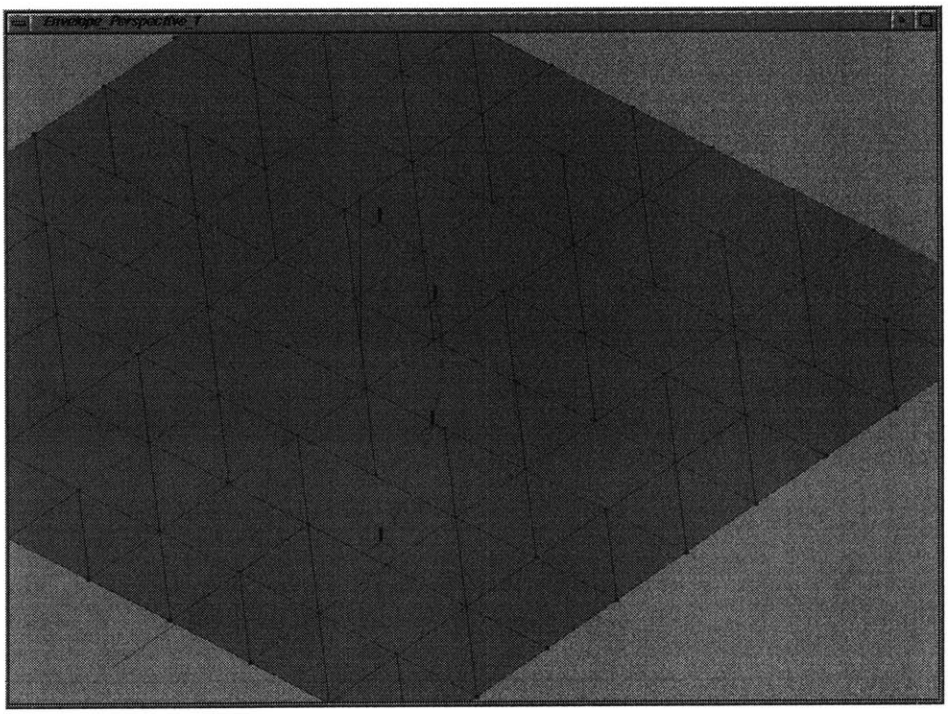


Figure 5-3: Curve-On-Surface as input Section Sweep Tool.

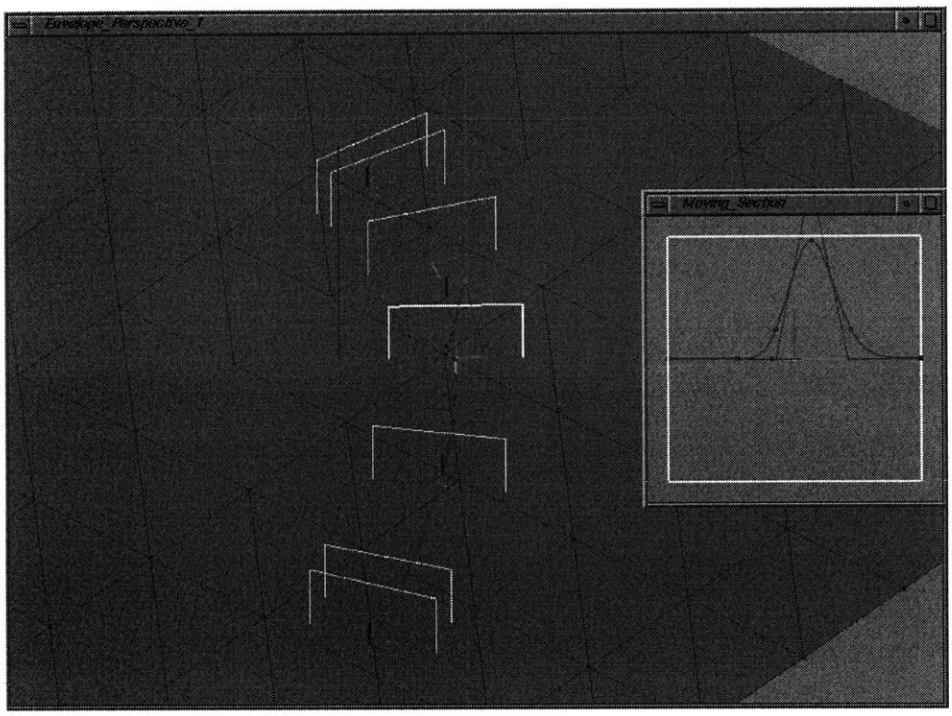


Figure 5-4: Sections defined in the Section Sweep Tool. Note the inset window showing only a narrowly clipped section – the currently active section.

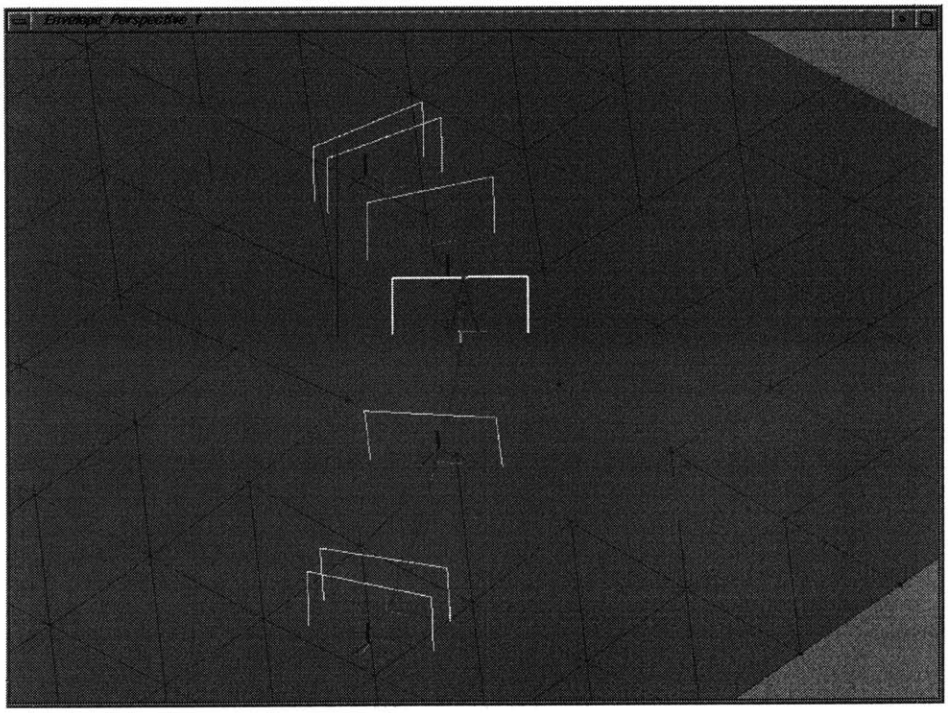


Figure 5-5: Result of the above edit using the Section Sweep Tool. Clearly, the underlying mesh density was insufficient to capture the detail that would be expected given the sections as defined.

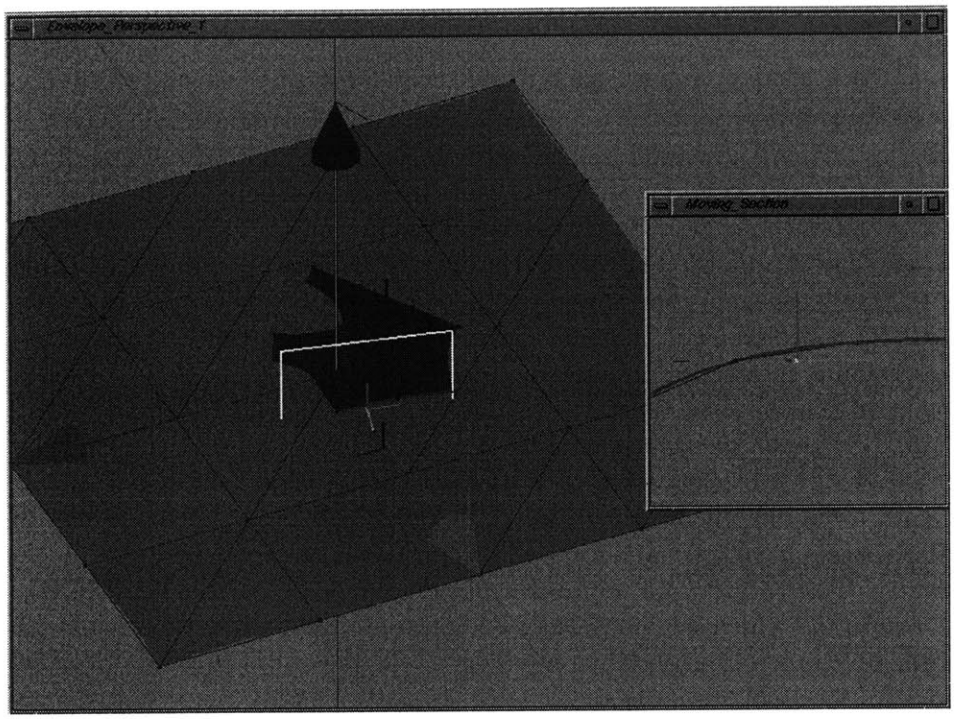


Figure 5-6: Snapshot of the cross-section as it is begin dragged along the curve

5.2.2 Dragging Sweep Tool

While the Section Sweep Tool is very useful for precision work, the Dragging Sweep Tool comes into its own as a tool for exploring different designs quickly. It should also be noted that this tool could be the focus of a great deal of future work. The main difference between the Section Sweep and Dragging Sweep Tools is how the sweep surface is defined. As the tool's name implies, with the Dragging Sweep Tool, the user actually drags out the sweep surface interactively (see Figure 5-6) using both hands, just as she would when modeling in clay.

Due to the limited interface available (a standard monitor, a keyboard and a mouse), certain simplifications were necessary in this implementation. However, there are many possible ways of improving the interface.

In this implementation, specifically, the user again places a curve on the surface in order to define a path for the sweep. Once the tool is entered, the system makes certain choices concerning how fast the drag will occur. In other words, when a modeler drags a rake across a clay surface she controls the speed of the drag as well as the angle of incidence of the rake edge. In the case of a blade, the modeler has additional control over the curvature of the blade itself.

In this implementation, I only have a very limited number of input channels available to me, and I therefore chose to hardwire some reasonable choices into the system. A more complete system would permit the user to modify the choices (which the current one does

not). Optimally, though, it should let the user make those choices dynamically, just as the clay modeler would.

Given the sail's velocity, the system still needs other defaults. Within the moving section plane, the cross-section is modeled as a single span cubic B-spline curve. Thus, it has four control points. The default configuration of these is collinear and in the line of intersection between the section plane and the surface tangent plane at the point on the curve-on-surface. This is a reasonable choice, but by no means an unique one.

The important aspect of this tool is the way in which the curve is deformed during the drag. In this implementation, the curve is deformed in response to the user's pressing various keys. Specifically, given that the user will need at least one finger to control whether or not the section is moving, symmetry dictates that I have four fingers on each hand with which to control the shape of the curve. I assigned the left-hand *home* keyboard keys "a", "s", "d" and "f" to the deformation of the left side of the tool and the right-hand *home* keys "j", "k", "l" and ";" to the deformation of the tool's right side.

Mapping a force increment to each key, I designed the forces so that when all four left hand keys are pressed, the left hand force will be maximal. (Note, that due to the limitations of this interface, the input is far from continuous). At each point when the cross-section "stops", I determine which left home keys are pressed and apply a downward force to the leftmost control point proportional to number of left hand keys pressed. Similarly for the right hand.

A number of limiting assumptions have been made in the above due to the small number of input channels I have at my disposal:

- the user is assumed to always want to create a convex sweep surface – if a concave sweep were needed, the force should be applied to the inner control points or the direction of the force should be reversed. A toggle sets up the tool for creation of a concave modification.

- the size of the force that is to correspond to each pressed key is a system defined choice – more properly, this should be determined dynamically from the user's actions. A continuous interface would address this as well.

- the curve is currently always anchored about some point (I chose the live point on the curve on surface) but the designer will generally want to change that dynamically. I attempt to support this using the arrow keys – but using them may force the user to let go of a pressed home key.

Nevertheless, even with the limited interface available, the tool is useful and clearly has great potential (see Figures 5-7 and 5-8). As mentioned, this Tool comes in two versions. Figures 5-7 and 5-8 show results of the use of the mesh replacing version of the tool.

5.2.3 Shape from Shading Tool

The third major tool in the implementation applies a machine vision algorithm. This algorithm was introduced by Bichsel and Pentland [BP92] as a contribution to the area known as *shape from shading*. Shape from shading is generally concerned with the production of

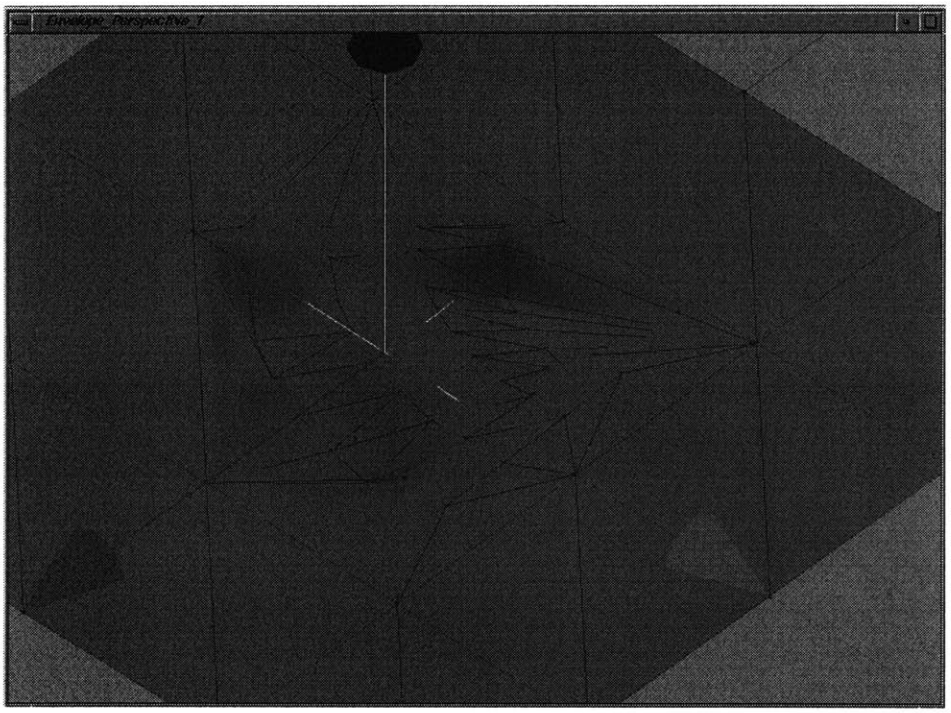


Figure 5-7: Resulting surface

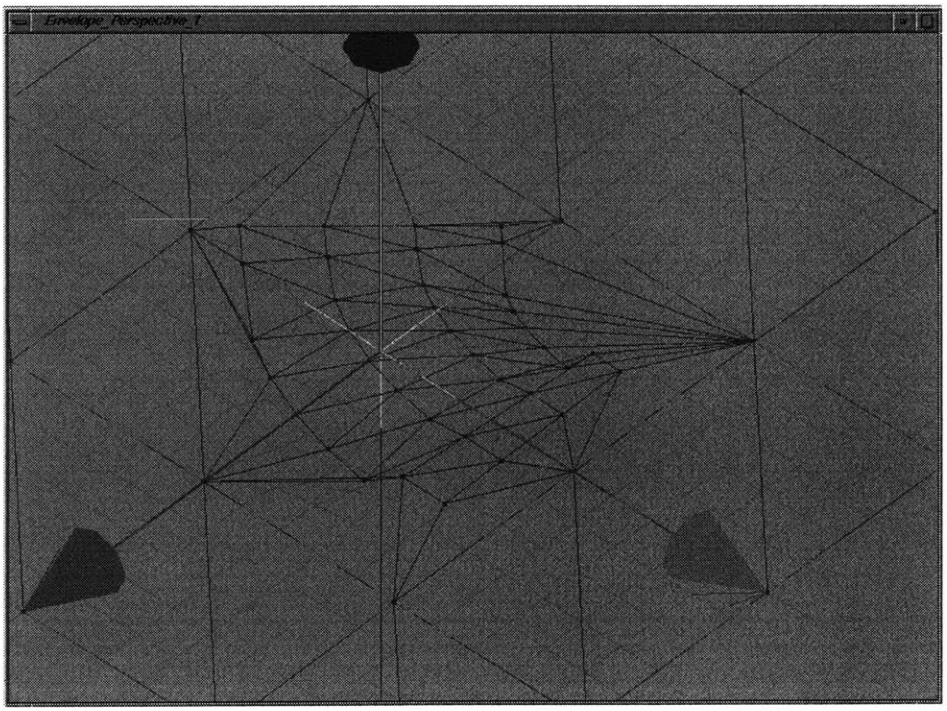


Figure 5-8: Resulting mesh

three-dimensional geometry based solely on acquired optical image data. This is a fundamentally complex and intractable problem as a single image could generally have been produced by an infinite number of scenes. In other words, the transformation that produces an image does not have a unique inverse.

As a consequence of this, researchers have concentrated on formulating reasonable assumptions that can be matched in real-world scenes so that algorithms can be created which attempt to find plausible inverses for that image generation transformation. Research in this field has produced algorithms which produce geometry based upon acquired stereo images (two simultaneously acquired images from different view points) by mathematically modeling the parallax between the images. A great deal of effort in this area has gone into developing accurate material models so that intensity values in the image can be interpreted correctly. Many have also developed algorithms which attempt to produce range data (height fields) based upon a single acquired image. The algorithm used here is an algorithm of this latter type.

The Shape from Shading Tool has been developed in response to the designer's need to be able to easily convert sketches into three-dimensional form. In a way, it is an extension of the Sketch Projection Tool. The essential idea is that a designer selects a scene (positioning the camera in a well-defined lighting situation) in which the surface region to be modified is well-displayed. Then, after an image of the scene is acquired directly from the screen buffer, the designer sketches directly in that image. In the final step, the modified image is compared to the original and a height-field capturing the changes is created.

The success of most of these techniques depends squarely on how well the materials in the scene are modeled. In the Shape From Shading Tool, we have a significant advantage, because we explicitly control the scene itself. Therefore, we know the shading model precisely. Also, to the extent that the modified image is unchanged, we also know a great deal about the scene itself (i.e., for a given pixel location, we know the exact entry in the height field corresponding to the original image).

As part of this research, a prototype of the tool has been implemented. In essence, the user sets up a view and picks the surface to be edited in that view. Internally, the image in the screen buffer is captured and written to disk along with a copy. An image processing tool, in this case the *GNU Image Manipulation Program* (a.k.a. *The Gimp*) [Bun00], is launched on the image. Within the image manipulation tool, the user is able to use any tools she likes to create a new shading pattern for the surface in the view. When she is finished, *The Gimp* is exited and the system compares the modified image against the copy that was saved initially. In a direct application of the Bichsel-Pentland algorithm a height field for the modified image is created.

Similar to the final step of the simpler versions of the two sweep tools, mesh vertices in the original surface are moved appropriately if, in the new height field, they are given a new height. Although it has not yet been implemented, it seems clear that this tool might also benefit greatly from a second version in which a new sub-mesh may be substituted in place of regions of the original mesh.

5.2.4 Mesh Replacement

As described earlier, both the Section Sweep Tool and the Dragging Sweep Tool exist in two versions. The first and simpler version is as described above. The second, far more powerful versions of each of these tools involves an important change. Where the standard version merely moves affected existing mesh nodes to the "sweep" surface defined by the user defined or dragged cross-sections, the second versions actually involve replacing the affected mesh region with a better mesh.

For the "sweep" surface that the user creates, the finite element mesh created is defined by the cross-sections as defined in the tool.

The next step is the identification of mesh vertices that would be moved. Next, all elements that make use of those vertices are collected. For each boundary edge of the element set, we check if any points along it are being affected. If any points on an edge would be moved, we add the neighboring element to the set. We continue this until no more elements need to be added. The mesh nodes and edges along the boundary of this set are locked down so that surface deformations inside the boundary will not affect the surface outside. In effect, this is the type of local control that a freeform surface designer expects.

The elements that are inside the region are deleted from the finite element mesh and the elements from the new mesh are added in their place. At this point, we have yet to fill the gap between the original region boundary and the outer boundary of what is the new mesh.

Completing this merge requires a surprisingly intricate process, consisting of essentially three steps:

- Geometrically determining the topology of the required gap-filling mesh. For the general case, this is an astonishingly intricate task.
- Embedding the interior mesh in the domain of the original surface. Setting up a robust particle system [Ree83] to determine good parameter values for the nodes in the interior mesh that are consistent with the known parameter values on the boundary.
- Ensuring fast solution times for the finite element surface representation of the resulting mesh. Applying a bandwidth reduction algorithm to the numbering of the final mesh's degrees of freedom.

Each of these topics will be treated in turn.

Determine Topology of Gap-Filling Mesh

Consider the very simple configuration shown in Figure 5-9. Let us define the *outer boundary* in this configuration to be the one comprised of the solid nodes. Thus, the outer boundary is really the boundary of the hole that is to be cut out of the original mesh. We call the boundary defined by the clear nodes the *inner boundary.* The inner boundary is the boundary of the new mesh that is being inserted in place of the hole.

O'Rourke has described techniques to create triangulations of planar polygons [O'R94]. His descriptions do not take into account the need to produce triangulations with elements having good aspect ratios nor does he directly address filling in a polygon defined by two

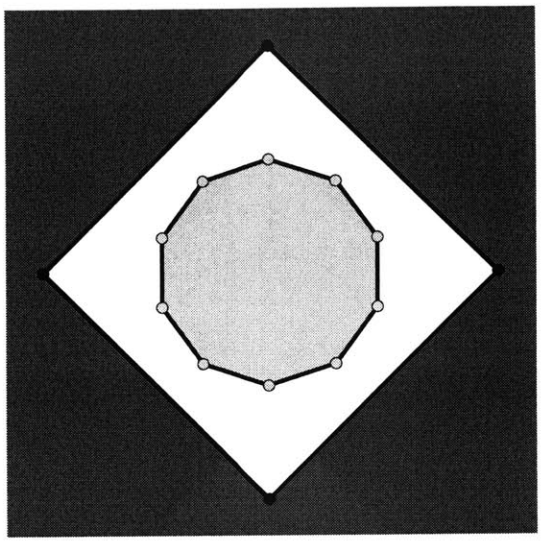


Figure 5-9: Configuration of gap that must be filled.

separate boundaries. Of course, I could simply add one well-chosen special-case edge to get around that problem. The methods described by O'Rourke do have the benefit that they work for non-convex hole boundaries. This is an area where more work needs to be done and a hybrid algorithm should be developed to handle the most general case while producing well-shaped triangles.

In the implementation, we assume that we have both of the ordered boundaries (in terms of the nodes). By convention, in this implementation, a boundary is defined so that elements which will be retained are on the left as one travels along it. Thus, for the purposes of what follows, we will assume that the order of either (but not of both) of these boundaries has been reversed. Thus, in the diagram both boundaries may either be traversed in a clockwise fashion or they will both be traversed in a counterclockwise one.

The first step is to determine, for every node on the outer boundary, the nearest node in the inner boundary (these associations are shown in dotted lines in Figure 5-10). These node pairings are collected and then checked to make sure that for any two sequential nodes on the outer boundary, the corresponding inner boundary nodes are not out of sequence – although they could be the same node. If any inner nodes are out of sequence the pairings are modified until the sequence is correct – even though, in general, the inner node associated with a particular outer node in a pairing will no longer be the nearest inner node.

Each node pairing will produce an edge in the final mesh so we can go ahead and add those edges. Next, the outer boundary must be traversed, two nodes at a time, and node pairing examined. Using a heuristic algorithm targeted at achieving triangular elements which have good aspect ratios and are oriented correctly, the resulting gap is filled as in Figure 5-11.

The above is clearly a simplification of the actual algorithm. For instance, it should be clear that there are multiple cases of gap sub-region configurations (see Figure 5-12).

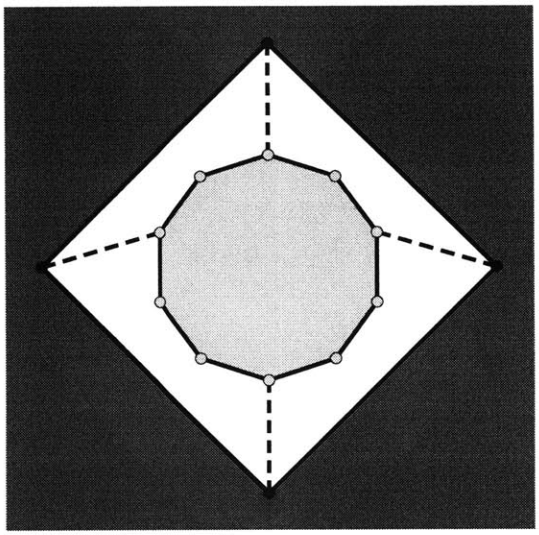


Figure 5-10: Node pairings indicated by dashed lines.

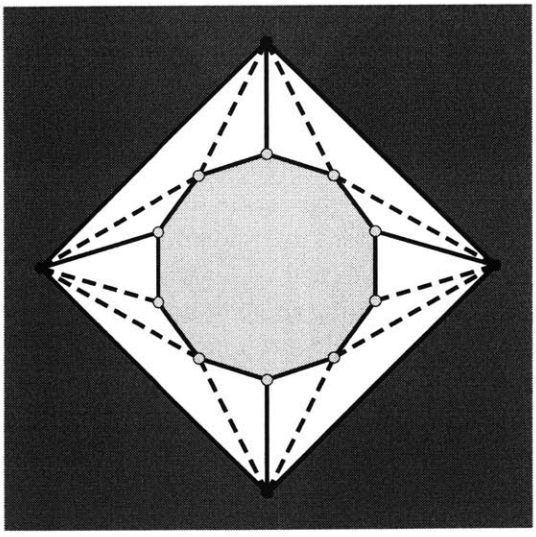


Figure 5-11: Fill in the sub-regions in the gap.

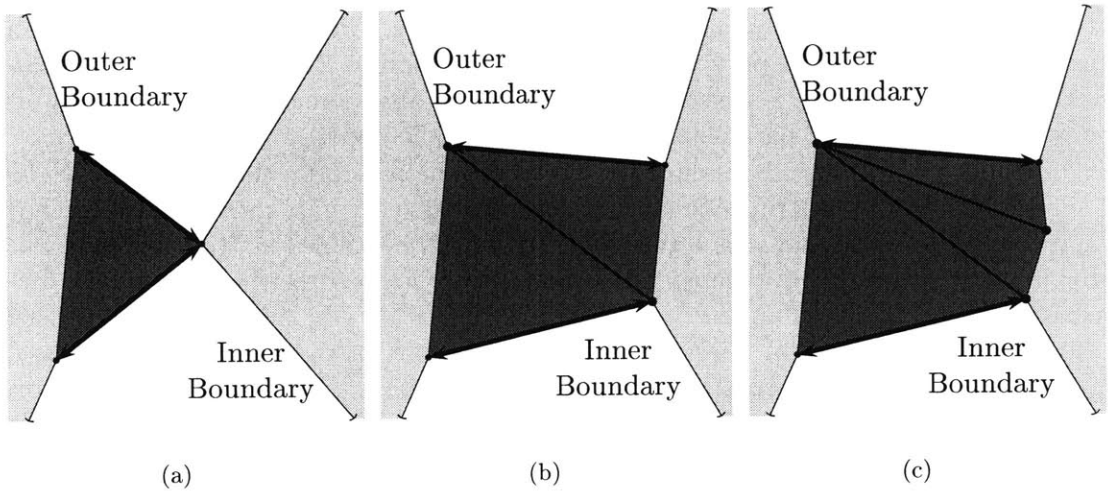


Figure 5-12: The three major gap types.

Actually, there are even more cases which often arise when either of the boundaries changes direction sharply relative to the other.

The result of this step is a topologically-correct filling in of the gap between the surfaces. Although the nodes in the swept sub-mesh have a domain embedding that is consistent for that surface on its own, an embedding in the parameter domain of the original mesh is required.

Embed New Mesh Nodes in Original Domain

At this point, we have an outer boundary defining the hole that has been created in the original mesh. Working from the assumption that the original mesh had a satisfactory embedding in its domain (i.e., that the parameter values assigned to each node in the mesh produced nice surfaces), we need to determine parameter values for the newly added nodes that are both consistent with the original embedding and also produce a nice merged surface.

Clearly, there may be many ways to accomplish this depending on the definition of *nice* in this context. We know that, if possible, we need the newly assigned parameter values to avoid creating triangles which are flipped in the domain. That is to say, if the existing triangles are all defined, say, in clockwise order with upward pointing normals (in the domain), then we require the same of all newly created triangle embeddings. They should be *consistent* in this way. However, we really require much more than such simple consistency.

We also would like triangles which have relatively large surface area in three dimensions to have relatively large domain surface area. Similarly, triangles with small range areas should have small domain areas. In addition, we would like to see a preservation of the proportions between pairs of triangle edges.

This is a surprisingly challenging demand. To meet it in this implementation, a new

approach to the problem is utilized. The seminal idea is to construct a spring-mass system in the domain which properly captures these requirements. Once the system is constructed, it is released and, using a particle system with an implicit Euler step, the rest state is determined. If the system is constructed properly, the final domain position of each of the free nodes defines a good parameter pair to assign to the node.

Only nodes in the outer boundary and the nodes that comprise the new sub-mesh being added are involved in this process. Springs are constructed which correspond to each edge in the new swept mesh and in the gap-filling mesh. As mentioned above, the nodes on the outer boundary are already embedded and these are used as the fixed boundary for the spring-mass system.

For the nodes and edges on the mesh created as part of the sweep operation, several important facts are known. At the time of constructing the sweep surface which interpolates the user-defined cross-sections, a sweep surface is constructed that would have resulted if the user had not modified any cross-sections. For each edge in this unmodified sweep mesh, its length is stored with the corresponding edge of the modified sweep mesh. This length is called *EdgeParamLength* even though it is strictly a three-dimensional length (note that edges added in the gap-filling step will not have this variable defined). This will be used to determine the rest length of the spring corresponding to this edge in the spring-mass system.

Looping over the edge, we initialize *EdgeLength* to be *EdgeParamLength* if it is defined or, as is the case for edges involving at least one boundary node, initialize it to the actual three-dimensional length of the edge. Next, for each edge, define

$$\begin{aligned} SpringRestLength &= ScaleFactor * EdgeLength \\ SpringConstant &= 2.5 \end{aligned}$$

The value 2.5 was chosen as a result of extensive tuning and is determined in conjunction with the system's step size. *ScaleFactor* is an attempt to capture the difference in scale between the parameter domain and the range for the region. In this implementation, a good definition of *ScaleFactor* proved to be

$$\begin{aligned} ScaleFactor &= \frac{ScaleFactorLong + ScaleFactorShort}{2} \quad where \\ ScaleFactorLong &= \frac{Longest2DEdgeLength}{Longest3DEdgeLength} \quad and \\ ScaleFactorShort &= \frac{Shortest2DEdgeLength}{Shortest3DEdgeLength} \end{aligned}$$

These measurements are taken for the boundary nodes as the interior nodes do not yet have meaningful domain lengths.

In addition to the Hookean springs modeled in this system, a dampening component is also introduced so that the system comes to rest more quickly. In the interest of speed and stability, the particle system simulation is run using Implicit Euler steps (strictly speaking, an explicit step is faster than an implicit one, but the increased stability of implicit steps

allows us to take much larger steps and thus, the overall time taken in the simulation is reduced).

Reduce Bandwidth of the System

The final step in merging a new sub-mesh into an already existing mesh is motivated, not by the need for quality in the ultimately produced surface, but rather by the need for speed in the surface's finite element system solution.

As has been alluded to earlier, ensuring that the stiffness matrix K has a very narrow bandwidth is absolutely critical to fast solution times. Solving the system of equations generated as part of the finite element methods involves solving a banded symmetric positive definite $N \times N$ matrix. Solving a dense $N \times N$ matrix takes $O(N^3)$ serial time whereas solving a banded system of the same size takes $O(N^2)$ time (where the bandwidth is assumed to be about $\sqrt{N}$) [Dem93]. Clearly, there is a big incentive to maintain a small bandwidth.

Assuming the original mesh had a small bandwidth, there is no absolute guarantee that simply adding the new degrees of freedom onto the end of the list will produce a narrow bandwidth. In fact, it most likely will produce a terrible one. In a finite element interpolant, if the numbering of the degrees of freedom in a particular element is such that the largest integer difference between any two dof indices is larger than, or as large as, that of any other element, that integer difference will, in fact, be the bandwidth of the system. Thus, the cause of keeping the bandwidth small, is converted into the cause of ensuring that assignment of indices to nearby degrees of freedom occurs in some optimal way.

Finite element analysis has addressed this question and a solution is formulated in terms of graph theory [Fan94, CM69, GWPS76].

Definitions From Graph Theory The bandwidth, β, of matrix, K, can be formally defined as

$$\beta = max\{\|i - j\| : k_{ij} \in K \; and \; k_{ij} \neq 0\}$$

Consider the degrees of freedom in the system as connected in a planar graph. Thus, an edge between two vertices in the graph represents the fact that the nodes are shared in an element. Let the vertices be $V = v_1, v_2, ..., v_n$. Formally, then, we have a graph

$$\begin{aligned} G &= \langle V, E \rangle \;\; where \\ E &= \{(v_i, v_j) : v_i, v_j \in V, i \neq j\} \end{aligned}$$

and $(v_i, v_j) \in E$ defines *adjacency* between the two vertices. For a given v_i, the *degree* of that vertex is the number of edges that use it. A *path* in the graph is an ordered sequence of connected edges. A graph is *connected* if, for any two vertices, there is a path between them. The *distance* between two vertices is the number of edges in the shortest path between them. For any pair of vertices in a connected graph, we have a distance associated with them. The longest distance between any two nodes in the graph is the *diameter* of the graph. Often the diameter is also said to be the path itself (not just its length).

Define a one-to-one map $f(v_i)$, called a *numbering* of G. (This corresponds to the method of assigning indices to the degrees-of-freedom). For a given f, we can define the bandwidth $\beta_f(G)$ of G

$$\begin{aligned} \beta_f(G) &= max\{\|f(v_i) - f(v_j)\| : (v_i, v_j) \in E\} \ \ and \\ \beta(G) &= min\{\beta_f(G)\} \end{aligned}$$

Level Structure A *level structure*, $L(G)$ of depth k is a partition of V into k levels such that

- all v_i adjacent to vertices in L_1 are in L_1 or L_2
- all v_i adjacent to vertices in L_k are in L_k or L_{k-1}
- all v_i adjacent to vertices in L_j, $(1 < j < k)$, are in L_{j-1}, L_j or L_{j+1}

A level structure is said to be *rooted at* v, $L_v(G)$, if

- $L_1 = \{v\}$
- for $i > 1$, L_i is the set of vertices adjacent to vertices of L_{i-1} but not yet in the level structure.

The number of vertices in a given level is that level's width, w_i. The largest width in a level structure is the width, $w(L)$ of that level structure. Consider a numbering, f_L, which simply assigns sequential integers to vertices as they are encountered while traversing the tree, beginning with the root vertex and visiting every node on each level before continuing to the next level. Clearly, the bandwidth for this numbering is given by

$$\beta_{f_L} \leq 2w(L) - 1$$

The smaller we can make $w(L)$, then, the smaller the bandwidth becomes. It is useful to note that, in general, the greater the depth of a level structure, the smaller the width.

Bandwidth Reduction Algorithm If we truly required the minimum bandwidth, we would approach this differently. It seems intuitively clear that the minimum bandwidth will be generated when the level structure is rooted at one end of the diameter of the graph (diameter in the sense of a path). It is not generally practical to find such a vertex within a reasonable time. Thus, we simply choose a vertex which has the smallest possible degree in G (there may well be many possible choices) and iterate towards a better root vertex. The iteration is accomplished as follows:

- choose any v with minimum degree
- generate a level structure rooted at v. Call S the set of leaves (the vertices in the deepest level). Order members of S by degree (smallest first)

- sequentially generate level structures for each s_i, where $s_i \in S$. If, for any s_i, the depth of L_{s_i} is greater than that of L_v, set v to be that s_i and return to the previous step.

- from the set of level structures produced by elements of S and by v, choose the level structure with the smallest width.

With that near-optimal level structure in hand, a numbering must be generated as follows:

- assign to the root node the first number (presumably 0).

- for each successive level below the first, order the vertices by two keys. First, according to which has the smaller number of adjacent vertices in the preceding level and second, according to the degree of the vertex in increasing order. Number the vertices on each level successively according to this order.

This implementation takes advantage of this bandwidth reduction algorithm. It is implemented according to Lian Fang's description [Fan94] of a hybrid Cuthill-McKee and Gibbs algorithm [CM69, GWPS76].

Results of Mesh Substitution

The mesh substitution variant has been added to both the Section Sweep Tool and the Dragging Sweep Tool. In fact, in the account of the Dragging Section Tool, this better version was illustrated. The figures accompanying that section (Figures 5-7 and 5-8) clearly show the effectiveness of this approach in capturing the user's modifications in the subregion.

5.3 Utility Tools

Along with the major tools described above, it was also necessary to implement a number of auxiliary tools in order to properly use the system.

5.3.1 Define Region Tool

By selecting a region of the mesh in any of a number of ways, including the creation of a closed curve on surface, the user is able to demarcate a subregion of the mesh and modify various aspects of it. For instance, it is often useful to define the boundary as having fixed nodes and edges. In this way a user can declare that changes made inside the boundary will not affect the surface outside of it and vice versa.

From within this tool, the user can also choose to delete a region from a mesh (allowing for trimmed surfaces) or to subdivide the mesh in that region so that subsequent tools have more mesh vertices to manipulate.

5.3.2 Create And Embed Curve-On-Surface

Many of the tools implemented here require a curve-on-surface as input. This tool lets the user pick points on the surface that are interpolated in the domain to produce a curve on the surface.

In addition to this, it is often useful to create a curve on the surface in order to embed it. By this, I mean that the user would like to first draw a curve on the surface and then modify the underlying mesh (without changing the resulting surface) so that mesh edges follow (in some way) the curve that was drawn.

5.3.3 Move Point-On-Surface Tool

A simple tool that lets the user rigidly move items in a set of pre-picked mesh topology.

5.3.4 Move Point-On-Surface Along Normal Tool

Closely related to the Move Point-On-Surface Tool, this allows the user to move pre-picked items perpendicular to the surface.

5.4 Other Miscellaneous Implementation Details

Where possible, I made use of efficient, freely available software modules. For instance, although I had originally implemented my own Gaussian Elimination routines, the final system actually takes advantage of the excellent LAPACK Fortran library. LAPACK, in turn, relies on the Level 3 BLAS library. As a consequence, a simple recompile would allow my software to take advantage of a parallel architecture (which my system does not have, at present).

At this point, the numerical aspects of this software have been very well tuned. In fact, the actual Gaussian Elimination and back-substitution steps require less than 20 seconds to solve a $10,000 \times 10,000$ system with bandwidth 250 although the overall time is greater due to time spent assembling the system. Further work on the assembly time is warranted. This timing is on a Silicon Graphics O2 workstation with a 180MHz processor and 192 Megabytes of RAM.

Also, this implementation has been largely ported to Linux (except the Shape From Shading Tool which currently relies on an SGI image utility library).

Chapter 6

Examples

As would be the case with any newly developed tools, it is important to establish that the tools implemented in this work actually enable designers to develop aesthetically constrained designs in a natural interactive way. Before giving an account of some such examples, it is important to acknowledge some existing impediments to such work. These impediments are not intrinsic to the system outlined here, but they are a reality in the current implementation – which is far from complete, especially where the user-load problem is concerned.

With that caveat, I have created two examples of the use of these tools. One example is an automobile door which was created primarily using the sweep tools. The second example takes advantage of the Shape-From-Shading Tool. Specifically, this second test is somewhat abstract in that it is not a direct design application: it verifies that the tool can be used to modify a surface using a photograph as the input sketch.

6.1 Automobile Door

This example began with a simple plane. Some initial work was done on the surface using the Move-Topology Tool as shown in Figure 6-1. Once this preliminary shaping work was complete, the real work began.

A curve-on-surface was placed on the surface along the top of the automobile door corresponding to the *belt line* (see Figure 2-24) of the automobile. Invoking the mesh-replacing version of the Section-Sweep Tool, I placed cross-section planes along the curve where necessary to capture the shape of the belt line. The cross-section curves are initially snapped to the shape of the surface in the cross-section plane making it easy to create a curve that is consistent with the underlying surface. This is particularly important at the ends of the curve where the modification surface (i.e. the sub-surface defined by the sweep curves) must smoothly blend into the original surface shape.

In the smaller inset cross-section window I sequentially edited the cross-sections curves to define the sweep surface form I was interested in creating along the belt-line (Figure 6-2). Once the cross-section work was completed, the resulting sweep surface mesh was created and automatically merged into the original surface.

Figures 6-3, 6-4 and 6-5 show some renderings of the resulting geometry.

The reflection-line renderings reveal certain minor imperfections in the way in which the

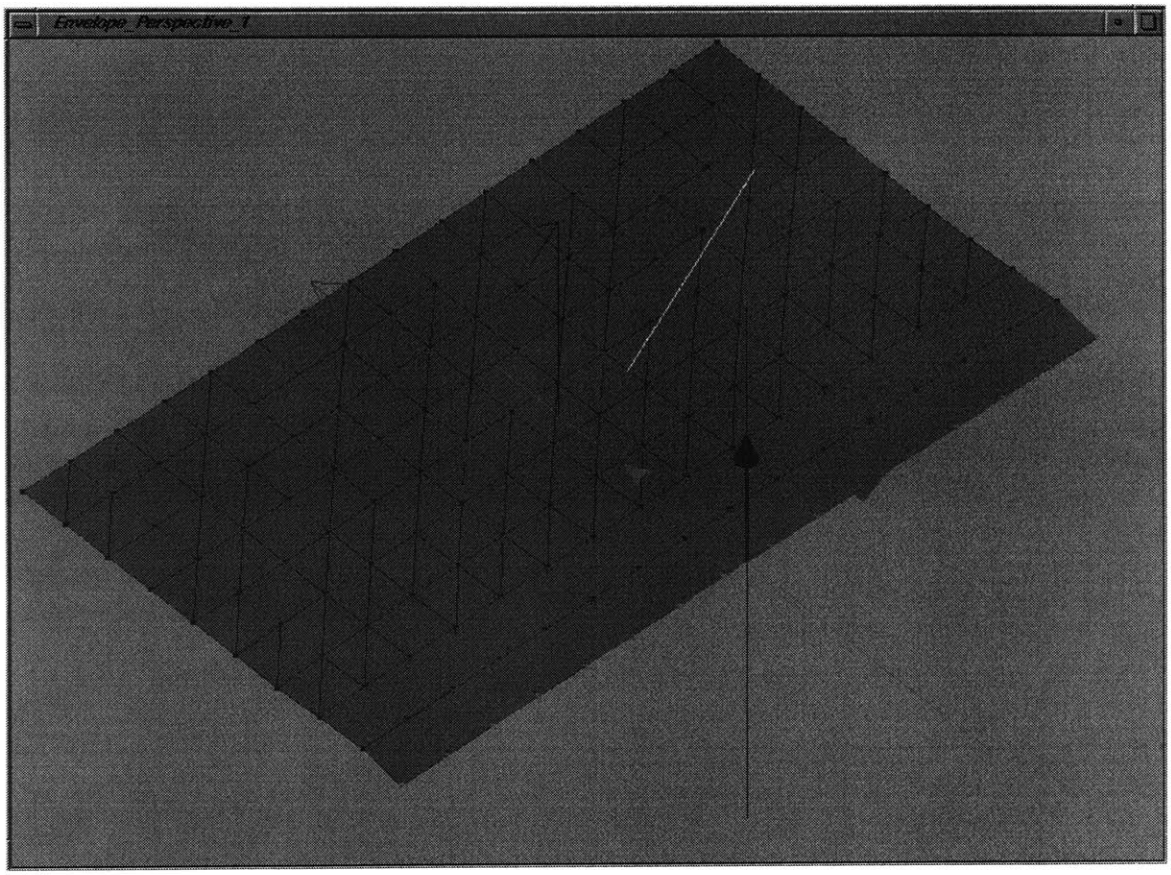


Figure 6-1: Using the Move-Topology Tool to move mesh vertices.

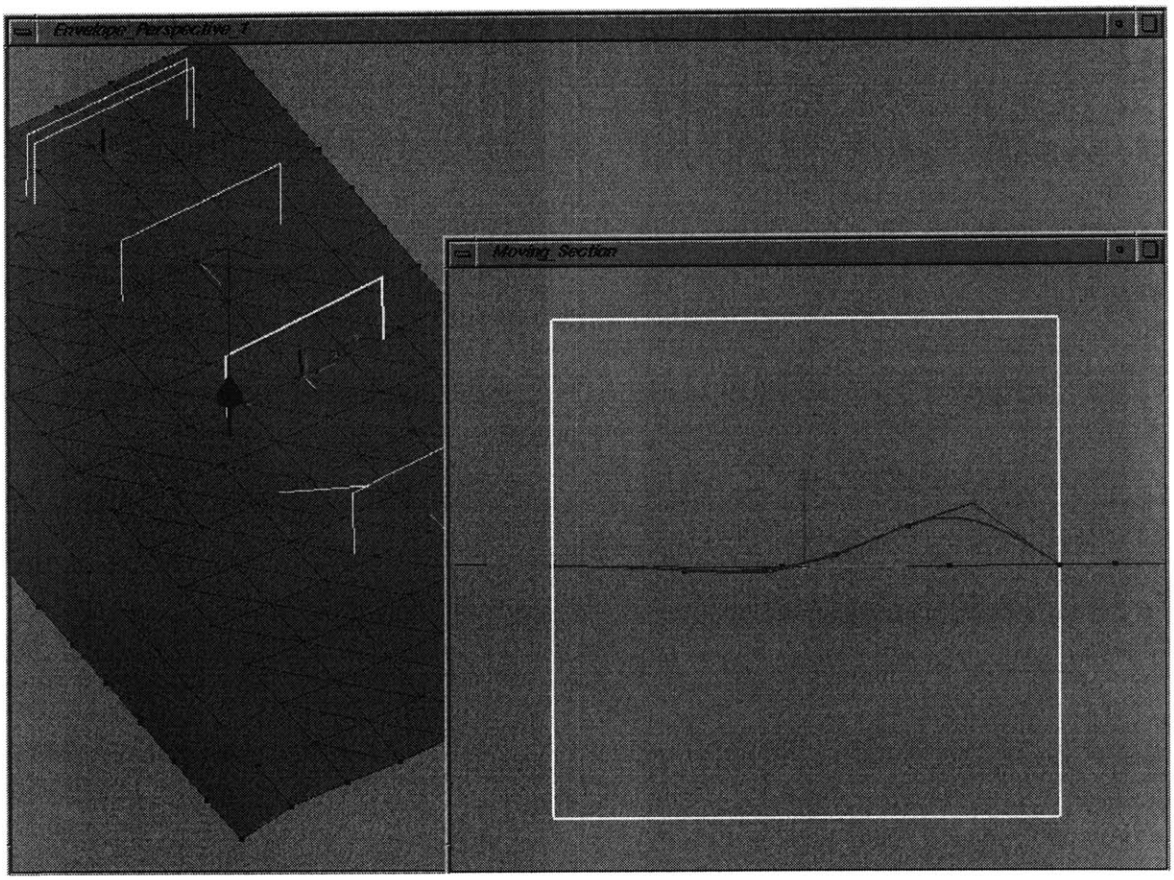
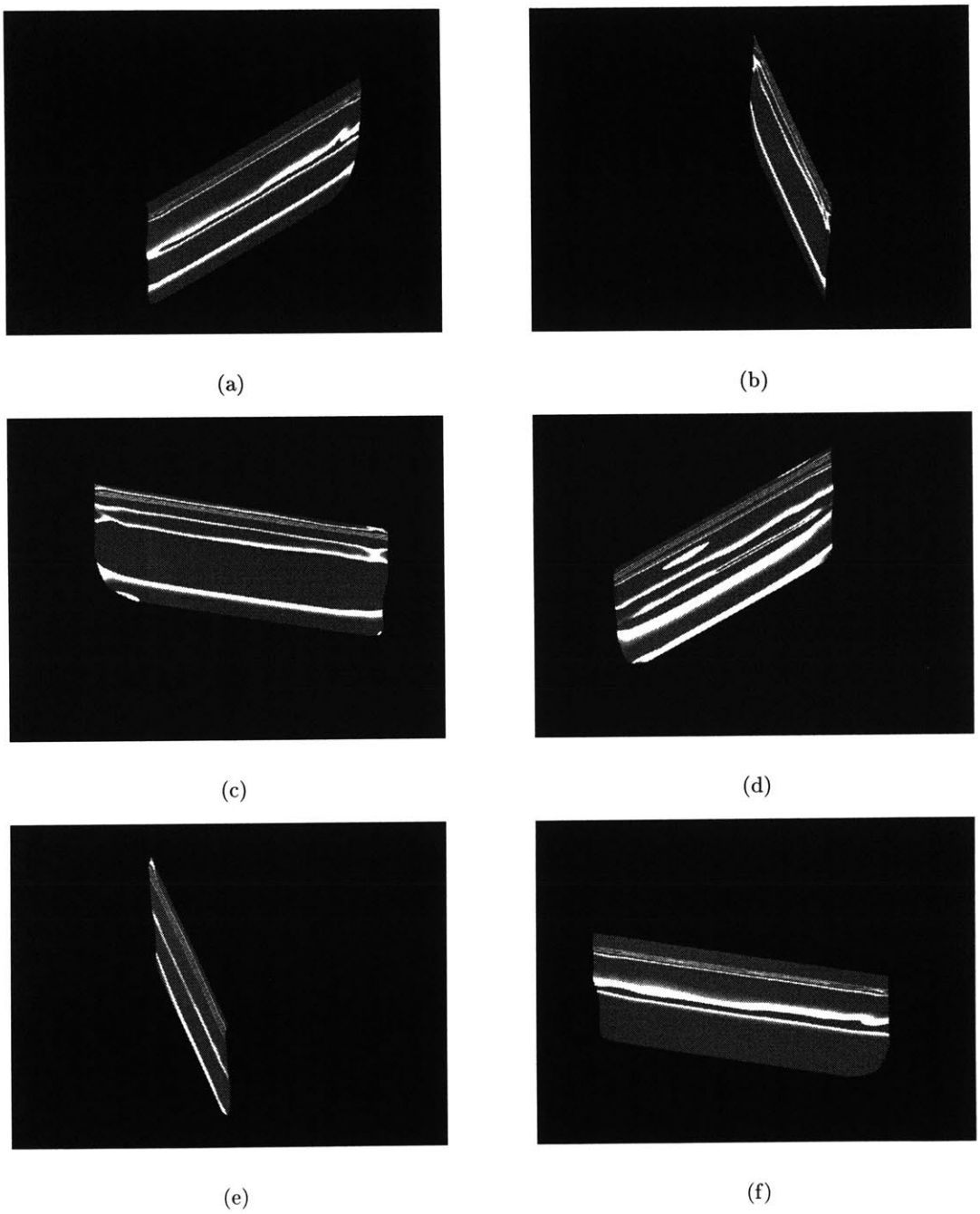


Figure 6-2: Cross-section curves are edited in the inset window.



(a) (b)

(c) (d)

(e) (f)

Figure 6-3: Frames from a turntable animation of the resulting car door surface.

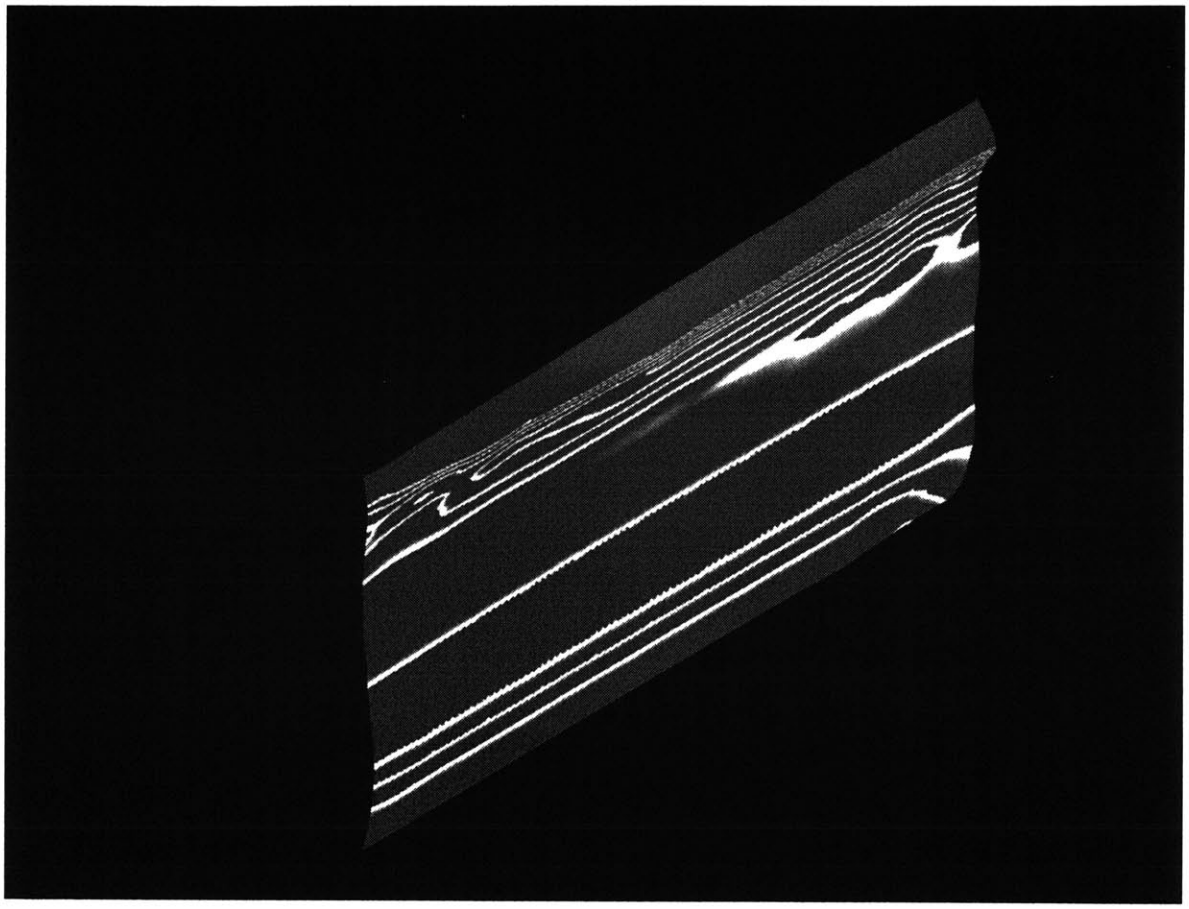


Figure 6-4: Rendering of the final car-door surface.

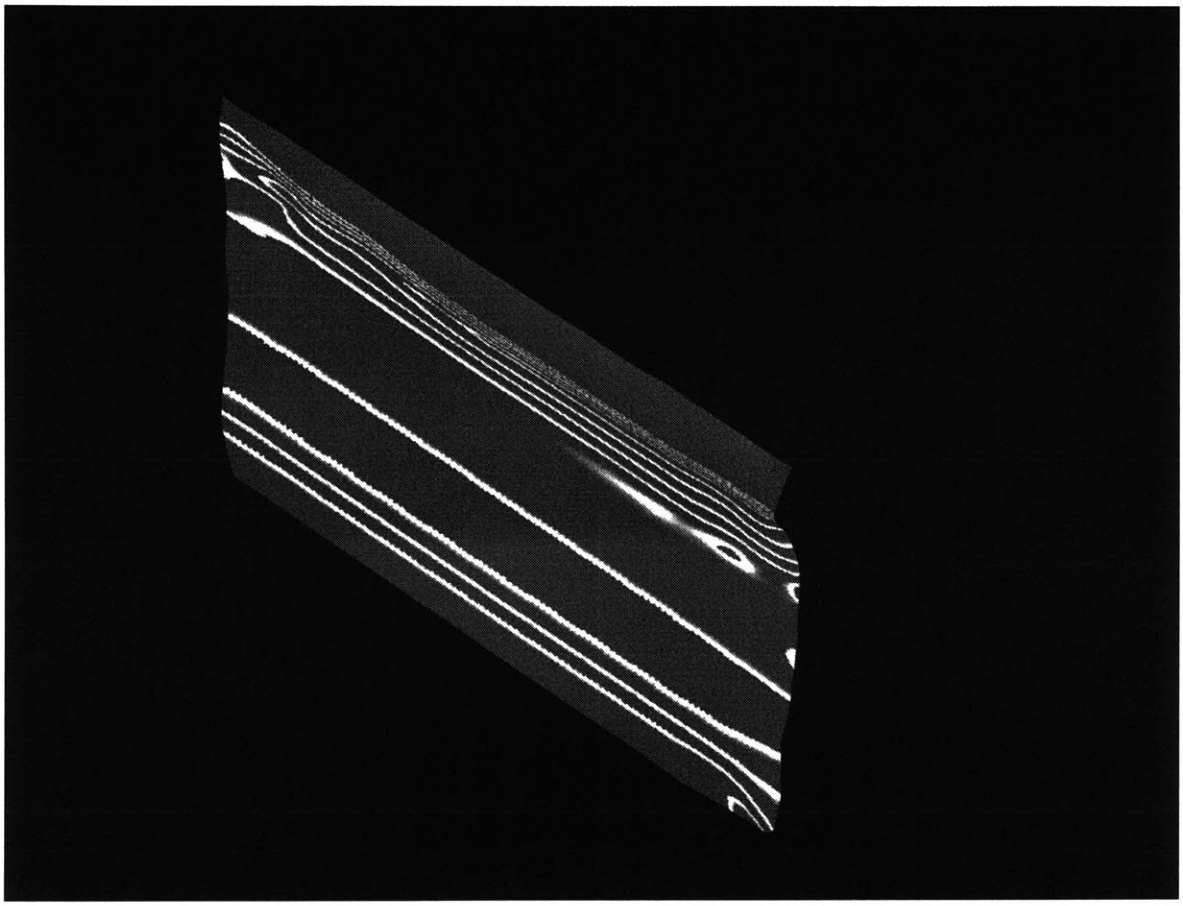


Figure 6-5: Another rendering of the final car-door surface.

new sub-mesh is inserted into the original mesh. It will certainly be an area of future work to improve the way in which that merging happens.

6.2 Happy Buddha

In this example, I tested the Shape-From-Shading Tool. This was a difficult task as the tool is implemented to meet the requirements, and more importantly, the talents of an experienced designer. The difficulty arose when I, an inexperienced sketch artist, sought to use the tool with my excessively crude drawings as input.

In an anticipated design process, a user would invoke the tool at some point in the process and use it to modify the design at that state. In effect, effective use of this tool is somewhat dependent on a preexisting surface being present. Due to the known user-load related obstacles to producing surfaces from scratch in this system, it is difficult to demonstrate such usage.

The key issue here is that the implementation is heavily slanted toward beginning with a good surface and modifying it within some sub-region. Specifically, the Bichsel-Pentland algorithm used to solve the shape-from-shading problem, can be initialized in two ways. The first, and more thoroughly discussed in their presentation, initializes the height field at the brightest point in the image and determines height values for the remaining pixels based upon that initial value. The second means of initializing the height field, is at the perimeter of some image region of interest. The latter approach is ideal for a situation in which the user wishes to modify a shape within a sub-region of an image in such a way that at the border between the original surface and the modified sub-region, the modified region of the surface smoothly blends into the original surface.

Clearly, if no particularly interesting original surface exists, the effectiveness of this tool will be limited. Another difficulty arises from the fact that this tool has no mesh-replacing version. Thus, the accuracy with which the resulting mesh will be able to capture the desired modifications is completely limited by the preexisting coarseness of the underlying mesh. In other words, the technology which captures the artist's idea of local control is not embedded in this tool – whereas it is in the two sweep-based tools.

6.2.1 Specifics of Happy Buddha Example

In this example I elected to test two separate aspects of the Shape-From-Shading Tool. The first portion of the test was to modify a plane base upon shading information in a photograph. Specifically, I began with a photograph of a very small white "Happy Buddha" statue (Figure 6-6). The lighting conditions in the photograph were carefully controlled so that the only significant light-source in the scene simulated a directional light coming from behind the camera (with no shadows cast by the camera or camera-man) and, importantly, producing minimal glare on the statue.

The resulting photograph was scanned and cropped so that only the Buddha face remained on an otherwise transparent image. Certain color correction was done while scanning to assure that the bright spots on the images were not saturated white – thus assuring a proper gradient across the face.

Figure 6-6: Original photograph used to create Buddha face surface.

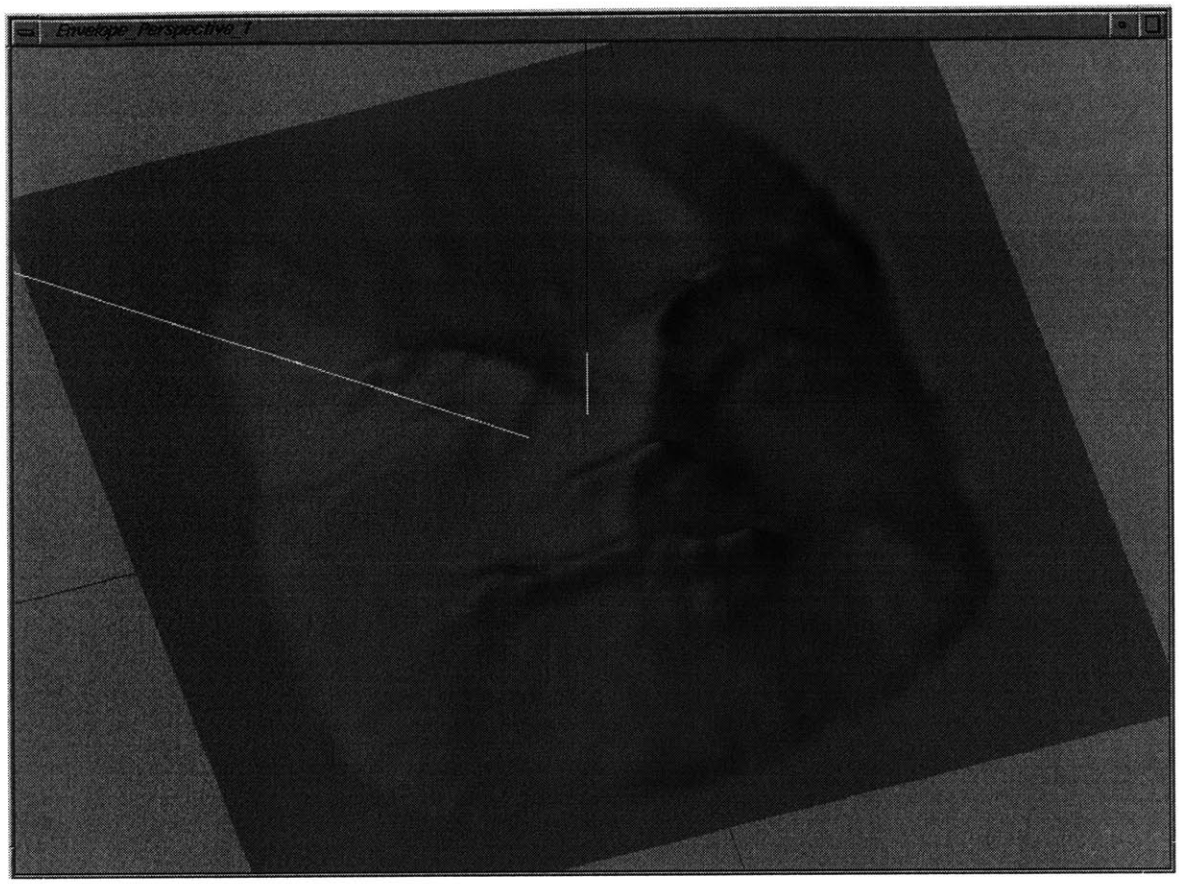


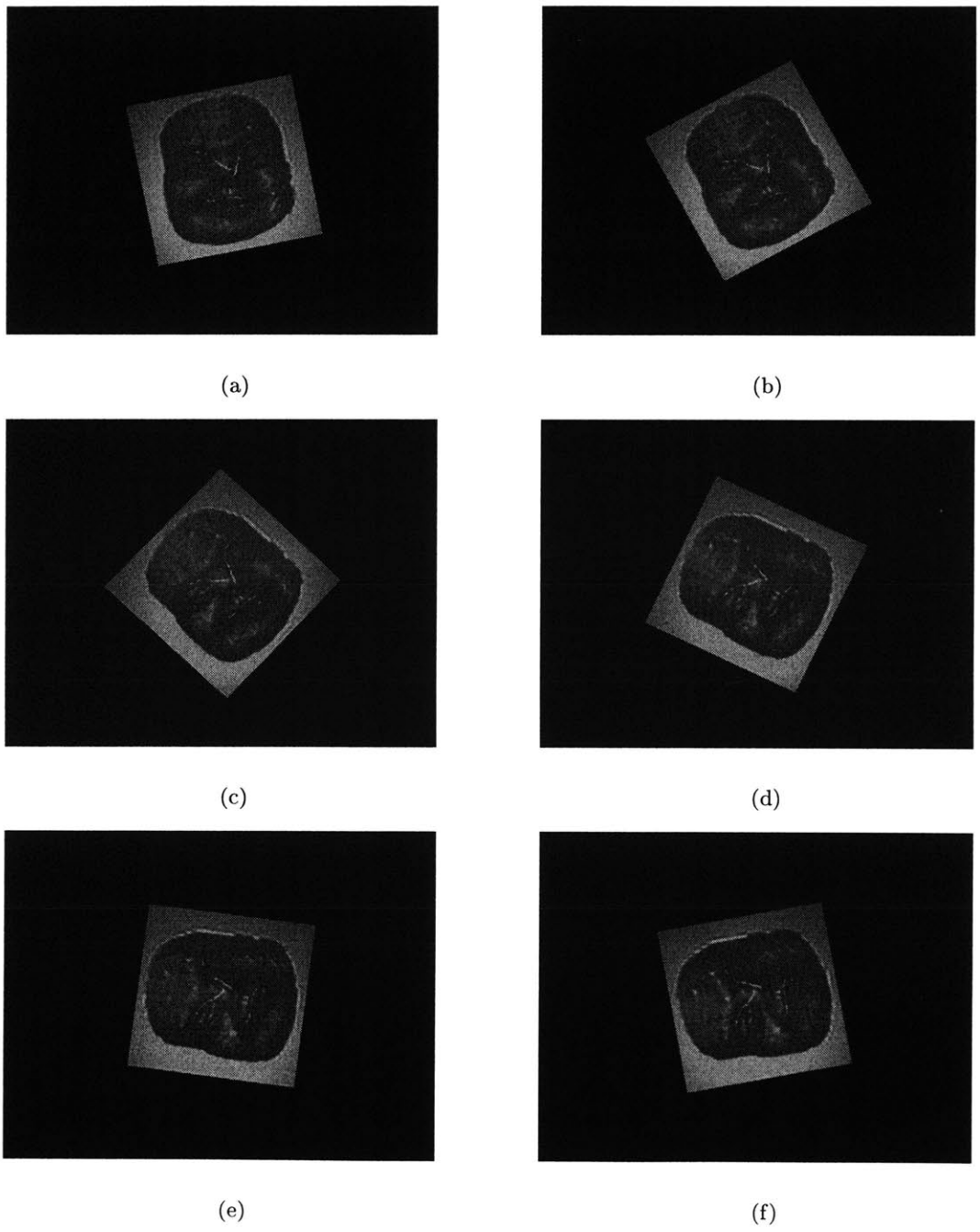
Figure 6-7: Screen capture of resulting surface incorporating Buddha face.

Creating Happy Buddha Face Surface

Within the prototype system, a moderately dense planar mesh (5000 triangles) was created and a head-on view of that mesh was established. The Shape-From-Shading tool was invoked on the mesh in that view and a grey-scale image was rendered and automatically opened in *The Gimp*. Within that image processing tool, the prepared Buddha face transparency was overlaid onto the surface image. Finally, that resulting composited image was saved to disk and the real work of the Shape-From-Shading Tool began. After some computation, the original surface emerged, but with modifications representing the information determined by the Bichsel-Pentland shape-from-shading algorithm. While this algorithm is able to produce a good height-field based upon the shading patterns it encounters, it has no means of establishing a scale for that field. Thus, the Shape-From-Shading Tool provides an interactive final step in which a user can adjust the scale simply by dragging the mouse. Images of the resulting mesh (Figures 6-7 and 6-8) shows that the system did an effective job of capturing the surface shape from the image. In addition, a number of images taken from an animation further show that the resulting mesh does in fact, resemble the original Buddha face (Figure 6-9).

Note, however, the uneven quality of the resulting surface. Although the shape-from-

Figure 6-8: High quality rending of resulting surface.



(a) (b)

(c) (d)

(e) (f)

Figure 6-9: Frames from a turntable animation of the resulting surface.

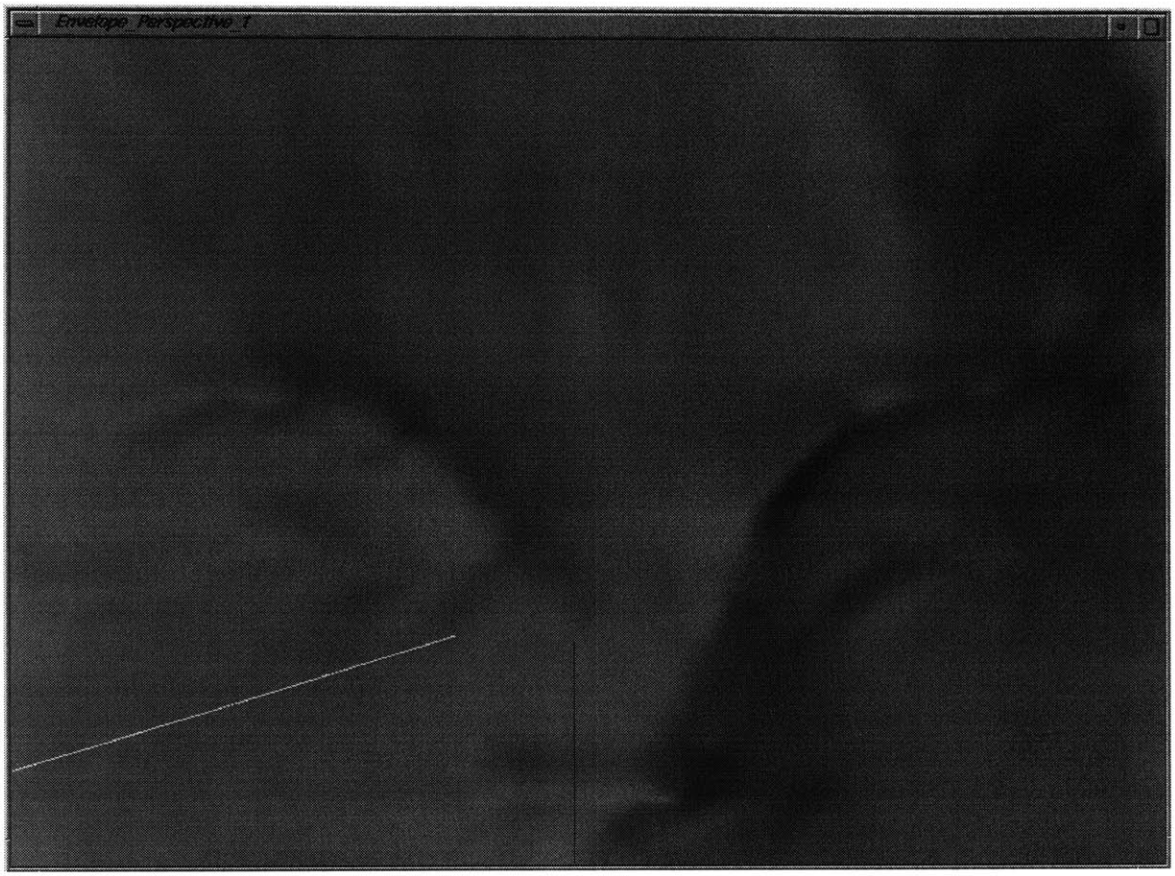


Figure 6-10: Screen capture of close-up of Buddha's forehead.

shading algorithm successfully inferred the general shape of the Buddha's face, there are slight undulations visible in the surface. This is an artifact due to the photograph's granularity and of the underlying mesh density rather than of the algorithm itself. Clearly, more work is warranted in order to develop techniques for minimizing the impact of the underlying mesh density on the resulting surface. One obvious approach would be to create a mesh-replacing version of this tool (much as I have done for the Section-Sweep and Dragging-Section tools). In the case of this tool, however, the situation would be more complex since mesh-density should be added only in those areas where it is required. We might also like to ensure that the mesh has edges along ridges in the image.

Modifying Happy Buddha Face

In order to verify the usefulness of the tool a as true surface modification tool, I applied it once more to the surface in order to add some detail onto the Buddha's face. I chose to design a furrowed brow for the Buddha.

Having setup the view of the Buddha's face that was suitable for this modification (see Figure 6-10), I again entered the Shape-From-Shading Tool for the surface in the particular view. In *The Gimp* I sketched a furrowed brow on the image of the surface (see Figure 6-11).

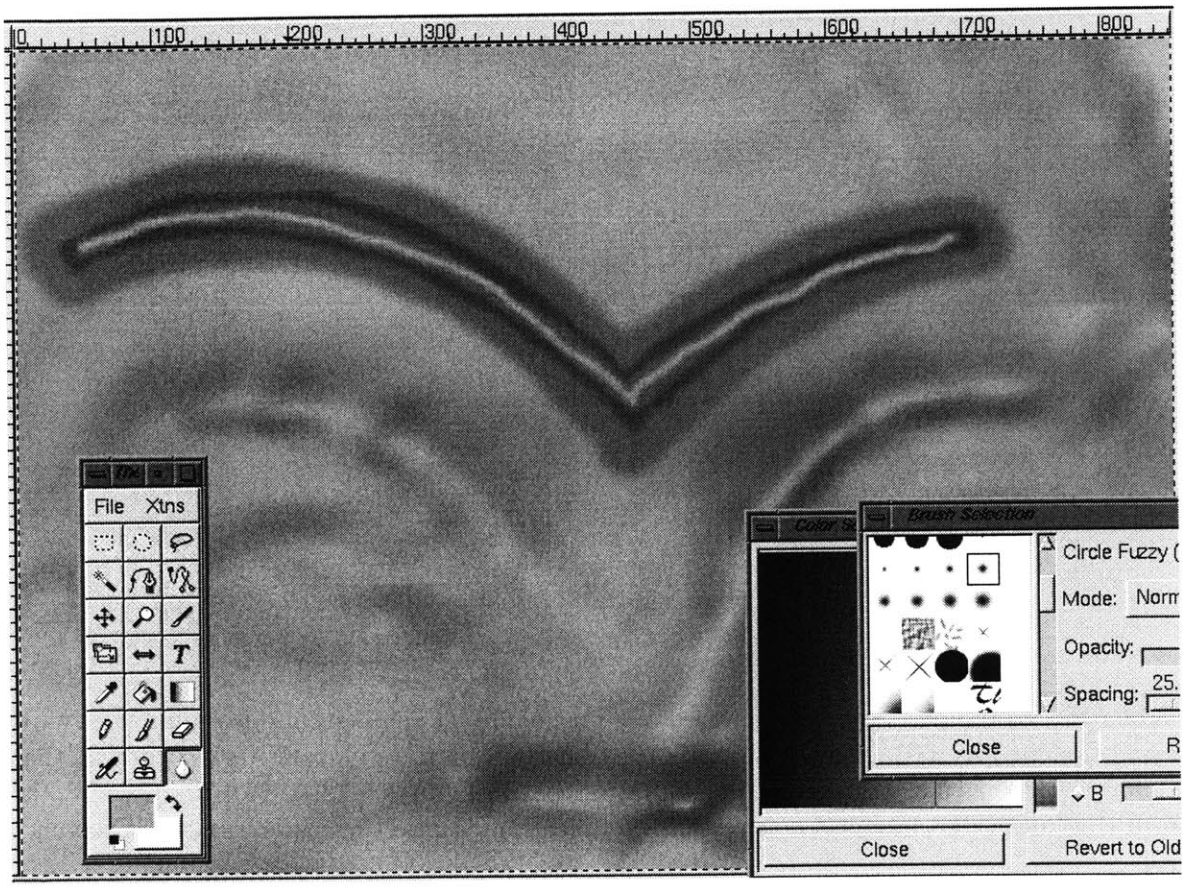


Figure 6-11: Screen capture of sketched image.

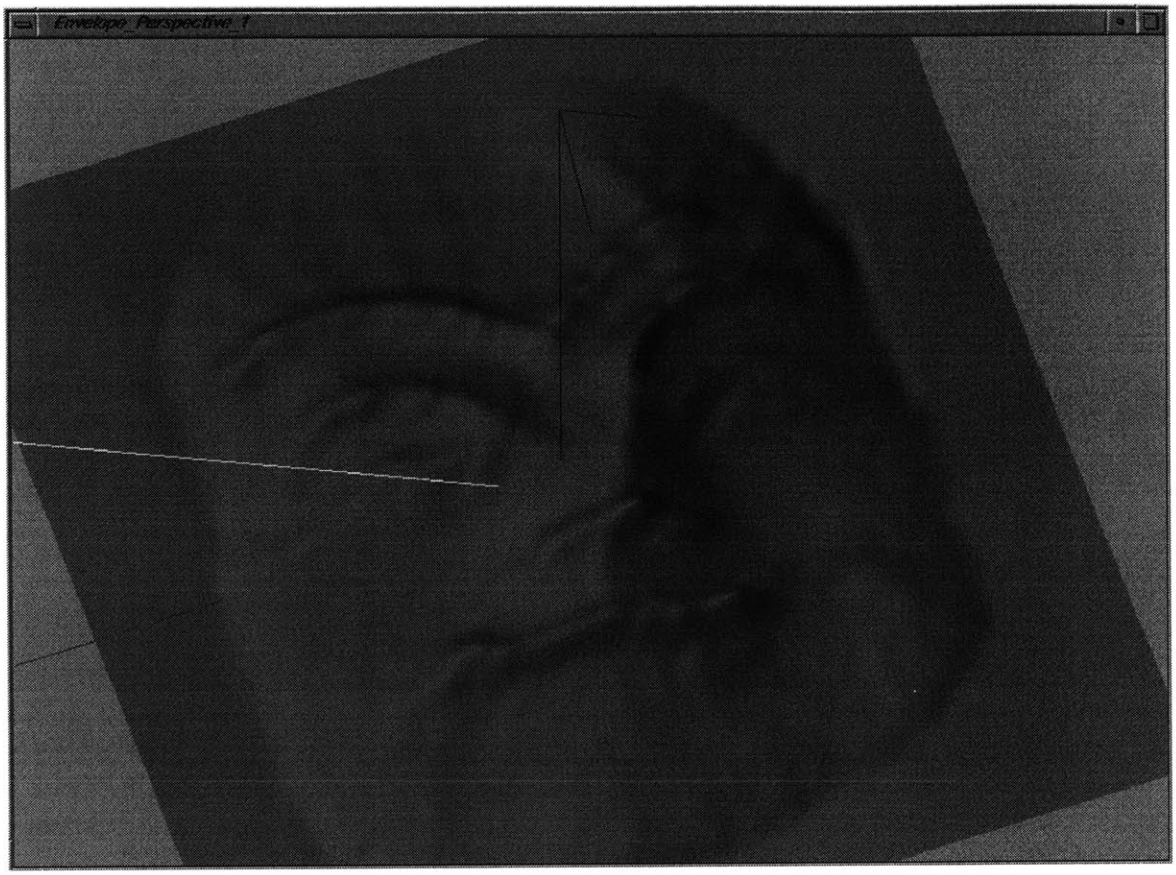


Figure 6-12: Screen capture of final modified surface.

When the sketch was complete, I saved the image and let the system determine the shape I had sketched as well as how to modify the underlying surface in order to incorporate that shape in that surface region. The results are seen in Figures 6-12 and 6-13. Clearly, the resulting mesh did, in fact, capture that modification despite the crudeness of my sketch.

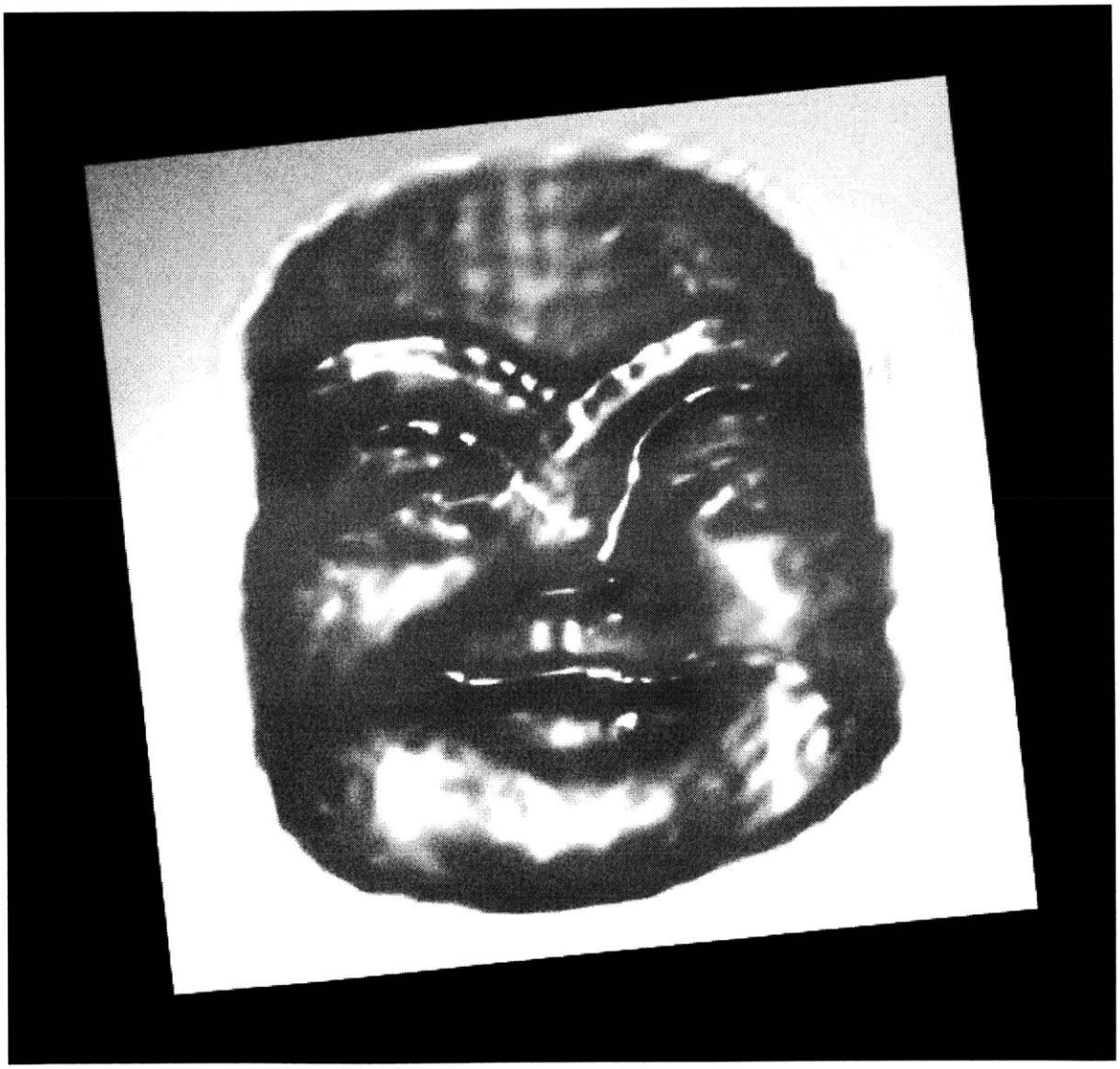


Figure 6-13: Rendering of final surface with modified brow.

Chapter 7

Conclusions and Future Work

7.1 Goals of This Research

The current application of computational tools in the design of freeform surfaces constrained primarily by aesthetic concerns is sufficiently limited that it begs the question: Why have creative designers shied away from using computational tools even though it would appear that such tools would offer them great advantages? In order to fully appreciate the reasons for this resistance, one needs to come to some understanding of the nature of their design processes.

Specifically, computational tools have emerged primarily from the engineering disciplines and an engineer's approach to design is significantly different from that of an architect or industrial designer. In engineering design, quality is primarily determined by how efficiently the design satisfies its requirements subject to its constraints. Aesthetically-constrained design is also influenced by such concerns of efficiency and effectiveness to a certain extent, but it is far more concerned with the appearance and meaning of the object being designed.

Beyond this fundamental difference in both design goals and approaches, there are other, much more practical differences as well. While engineers often sketch their ideas on paper early in their design process, such sketch work is at the core of the creative design process. In fact, this is a crucial observation. Specifically, the fundamental requirement of a new surface modeling system targeted at creative designers is that it support the same type of flexibility that designers find in sketching.

Thesis Statement: If computational tools are to be employed in the aesthetic design of freeform surfaces, these tools must better reflect the ways in which creative designers conceive of and develop such shapes.

What this means is that new tools should be focused on letting a designer change her mind easily as that exploratory component is crucial to her work. A tool that cannot support it must fail. In fact, this is almost certainly the reason that current computational tools are not used by most designers today: The cost of changing a design is sufficiently high that a designer is naturally discouraged from doing so.

The overarching motivation behind this work is to address this issue. This research seeks to begin the process of developing new tools that will enable designers to work naturally and comfortably in a computational environment.

Although this research has spanned a large number of disciplines, I undertook it with

a rather precise objective. Concerned about the difficulty involved in creatively designing a freeform surfaces in a computational environment, I set about developing new metaphors of interaction which would address the fundamental problems. It has been clear from the start that a productive attempt at addressing this problem would come only after a thorough study of the design processes meant to be helped by the new tools. Consequently, I determined that there were three important phases that this work must include:

- Examine current aesthetically-constrained freeform surface design methodologies.
- Develop a list of features that should be present in a software implementation if it is to address freeform surface design.
- Implement a proof-of-concept system to test some of the major items in that list of required features.

By addressing each of these three issues, more or less in turn, I have been able to take the first steps toward developing metaphors of interaction and even new tools that better support creative freeform surface design.

7.2 Results

As this stage of my research in this area draws to an end, it is important to assess the accomplishments to date. Not at all surprisingly, these results fall into the three categories established by the three phases of work described above.

7.2.1 Study of Current Creative Design Practices

As a starting point for this research, I needed to learn as much as possible about current freeform surface design methods as practiced in the field. The specific domains I intended study were architecture and industrial design. However, further analysis suggested that concentrating on industrial design – specifically automotive styling – would be more fruitful. As a group, architects have relatively little experience designing with freeform surfaces and as a result, there is limited collected knowledge on the subject.

A few architects, such as Antoni Gaudí and Frank Gehry, have embraced design with freeform surfaces; however, upon more detailed review, their design practices seem idiosyncratic and difficult to learn from. Gaudí's work involved actual sculpting, painstaking drawing as well as close site-supervision by the architect himself while Gehry's work is closely related to the work done by automotive stylists.

Thus, I spent several months studying model-making and design as practiced in BMW's styling department in Munich. While there, I worked on an actual project under the guidance of their model-making instructor. That process was seminal to many of the ideas that have been developed here.

Chapter 3 contains a complete account of my observations while at BMW. These observations included learning about the actual work of model-making in clay, watching the ways in which the designers and the model-makers interacted, and exploring ideas that emerged from my own discussions with designers and model-makers.

Specifically, this research provides a precise account of the different stages of the design process as well as the tools used at different stages, with emphasis on the design stages in which the designer and model-makers create a three-dimensional physical model.

7.2.2 List of Features for New System

From the work at BMW, a list of features integral to the design process emerged. These features must be present in a new computational system if the process I observed at BMW is to be transported into the computational realm. These features roughly correspond to actual design tools. I have categorized them into two groups: two-dimensional tools; three-dimensional tools.

The two-dimensional tools include tools that create three-dimensional geometry from two-dimensional inputs. A listing and brief description of these tools follows:

- Sketch Tool: It may suffice to take advantage of already existing sketching and painting tools, but given the specific characteristics of what is being sketched, it may be possible to create a more targeted paint system for sketching during the design process.

- Tape Tool: A user interface for the creation of tape drawings which, where possible, could ensure consistency between the various tape drawings as well as with package data. The specific and interesting quality of the tape that draws designers to this medium is that the tape has lateral resistance to bending which, in turn, helps them create smoother curves.

- Tape-To-Model Tool: Given the output from the Tape Tool or scanned tape drawings, it would be useful to have a tool that enables the designer to automatically (or almost automatically) create a starting model from which to work.

- Sketch Projection Tool: Given a rough model (perhaps the output of the previous step), it would be useful to be to project sketches onto the model so as to get a sense of the three-dimensional form. Perhaps, a library of basic car shapes would be collected so that designers could project their sketches long before they have created tape drawings.

In addition, this research identifies a suite of three-dimensional surface modification tools:

- Section Sweep Tool: Given an arbitrary curve-on-surface, the user places section planes along the curve and edits a cross-section curve within each section plane. The tool creates a sweep surface along the curve-on-surface which interpolates the defined cross-sections and then integrates the resulting sweep surface into the original surface.

- Dragging Sweep Tool: Similar to the Section Sweep Tool except that the section plane is dragged along the curve-on-surface as the user interactively modifies the cross-section during the drag.

- Shape from Shading Tool: A screen capture is made under known lighting conditions and the user then edits that captured image using the Sketch Tool or any image

processing system. Once the editing is finished, the system determines how the original image has been modified and attempts to deform the underlying surface so that it produces the sketched shading patterns under known lighting conditions.

- Define Region Tool: Supports local control from the perspective of the designer.
- Create And Embed Curve-On-Surface Tool: Many of the tools defined here rely on being able to define a curve on a surface.
- Move Point-On-Surface Tool: For a point on the surface, the tool lets the user move the point as she desires. This should be extended to let her directly manipulate the tangent plane at a point, as well as support similar logical extensions of such manipulations.
- Move Point-On-Surface Along Normal Tool: Similar to the Move Point-On-Surface Tool except that points are moved along the direction of the surface normal at the point.

Finally, a high-quality renderer will be necessary.

Somewhat orthogonal to the above list of features, other more general requirements were also generated from my observations at BMW.

- Virtual Reality Interfaces: Almost any of the tools in this requirement could be converted to permit a virtual reality interface – probably with great benefits for designers.
- Surface Representation: Properties of the surface representation should include:
  - Supporting a designer's definition of local control: The designer should be able to define an arbitrary closed boundary on a surface and define how the portions of the surface on either side of the boundary will interact.
  - Supporting global control: The designer must be able to propagate edits of the surface must be able to be propagated across the surface to the extent that locally controlled regions allow.
  - Supporting implementation of *level-of-detail*: Because designers require wide latitude (and not high quality) early in the design process and less latitude (and high quality) later in that process, it would be desirable if the underlying surface representation could operate in both of these capacities.

A system that implements these features will have gone a long way toward addressing the requirements of creative designers. Of course, these features together form a starting point and others could be added over time.

7.2.3 The Proof-of-Concept Implementation

In order to begin the process of verifying that these features do, in fact, work to define a more effective system for the creation to freeform surfaces, it was necessary to test some of them. The three-dimensional modification tools have been implemented in proof-of-concept form.

Specifically, the Section Sweep Tool, the Dragging Sweep Tool and the Shape-from-Shading Tool have been implemented and tested along with the remaining three-dimensional tools listed.

Choice of Surface Representation

Before embarking on the proof-of-concept implementation, it was important to set down some fundamental requirements for the surface representation:

- Support the designers concept of local control.
- Support global modification of shape.
- Support implementation of level-of-detail in design process.

An additional desired property of the representation is that it support a designer's direct manipulation of the surface. This distinguishes the representation from standard b-spline surfaces and almost all subdivision schemes. In the proof-of-concept implementation developed in this research, I chose to use the representation developed by George Celniker. In essence, this is a variational formulation designed to minimize a weighted sum of an area term and a blending term. I chose to implement this over surfaces using the finite element method.

Local control is a critical issue in its own right. Many researchers emphasize that one major advantage of b-spline representations is that such representations support local control. In this mathematical sense, local control merely means that moving a control point (or set of control points) will only affect a localized subregion of the surface. Thus, implementations can be extremely efficient in that they only need to re-evaluate those parts of the surface that actually change. Unfortunately, this mathematical notion of local control bears only the most superficial resemblance to what the creative designer thinks of as local control. The creative designer wants to design larger surfaces and then define sub-regions as locally distinct from the already defined larger surface. Thus they can work in the sub-region without affecting the already defined region outside the sub-region.

Of course, local control cannot come at the expense of global control. A designer will still want to make global surface modifications. In existing commercial systems, the designers may not harness global control directly as such modification is generally accomplished, if at all, by using some sort of global optimization method.

By choosing to use the finite element method to implement the surface representation, I was able to take advantage of the fact that many different finite element interpolants have been developed with varying properties. Specifically, some produce low-quality surfaces very quickly and others produce high-quality surfaces albeit more slowly. In most creative design processes, the earliest stages can be characterized as very volatile – making a fast surface representation desirable. As the design becomes more resolved, a designer will be willing to sacrifice some time in exchange for a higher quality surface. Thus, early in the process, a designer can choose a low-quality representation while upgrading to a higher-quality interpolant later in the process.

7.3 Contributions

The successful completion of this research undertaking has led to three important contributions – each corresponding to one of the three phases of the work.

First of all, a succinct account of the automotive styling process has been developed and documented here. This should provide future researchers with an solid starting point from which to depart on their quests to develop new and exciting tools for use by creative designers.

Next, by having translated my experience in BMW's styling department into a concrete list of features for a new system I have begun to formalize the requirements for such a system. The existence of such a list is a significant contribution in itself, but in addition, future researchers have a metric against which to judge the usefulness of their new tools.

The final major contribution is in the fact that I have begun the process of verifying the validity and merits of the feature list identified by this research.

7.3.1 Observations

Along the way, I have been able to make a number of observations which I believe help to clarify some of the reasons researchers have been somewhat slow to address the problems that are faced by creative freeform surface designers.

Flexibility

Flexibility is an extremely important ingredient of the creative design process. Any system that restricts a designer will understandably meet with resistance from design professionals. When translated into the parlance of the creators of computational tools, flexibility means the *ease of effective modification.* Relatively little research effort to date has been expended on creating powerful CAGD tools to let users quickly modify surfaces in a targeted way.

Interactivity

Other important conclusions may be drawn on the subject of interactivity. It has become something of a tradition to make claims of interactivity for new computer-aided design algorithms. According to conventional wisdom, new algorithms need to provide a designer with *real-time* feedback.

It appears that, despite the attention this demand has often received, such real-time feedback is not actually an issue or priority for designers. When discussing interactivity, this research has revealed that it is critically important to be precise. A system that is *interactive* at each of an enormous number of steps may be far less effective than a system which requires far fewer steps, regardless of whether such steps can be said to be interactive.

What really determines the effectiveness of a tool-set is the speed with which designs can be brought to a state of completion. Whether every step along the way is interactive or not is almost irrelevant.

7.4 Future Work

As many of the conclusions suggest, there is still a long way to go en route to a fully-functional aesthetically-constrained freeform surface design system. Nevertheless, the work so far suggests some important areas for further work. A significant amount of future work will address these user-load problem: in particular, the significant difficulty associated with creating useful starting geometry. Beginning with those aspects of the feature list for new systems that address the user-load problems, additional work must be devoted to implementing the remainder of the system, as well as to the development of other tool ideas. One such idea is to explore the possibilities of using techniques developed for gestural interfaces as another means of editing, and perhaps even creating, freeform surfaces.

An important area for future enhancement will be to add a parametric component to the implementation so that designers can return to design decisions made earlier without having to reproduce all of the work that followed.

Finally, future effort must be devoted to developing new and more effective physical user interfaces for freeform surface design.

7.4.1 User Load Problem

As a practical matter, in the implementation developed as part of this research, it has become evident that the major short-coming of this system is a result of the difficulty associated with creating new geometry. While the proof-of-concept tools in the application are well-suited to the modification of preexisting geometry, they are not at all suited to the creation of interesting shapes where none existed before.

While this is not a major problem for a proof-of-concept system, it renders the system near-useless as a full-fledged design tool.

Clearly, the Tape-to-Model Tool would go a long way toward enabling a designer to create useful initial geometry given a reasonable sense of the three-dimensional form as represented in two-dimensional projections.

In addition to the Tape-to-Model Tool, other approaches to generating initial geometry should also be explored. As described in this work, existing computational geometry techniques are extremely well-suited to the creation of three-dimensional forms – where the forms are created from scratch. However, it is not a trivial matter to convert output from standard CAGD tools to a format suitable for use as a finite element mesh.

One approach would be to convert a B-Spline surface to a mesh (of some precision) while preserving the domain information at each new node. However, it may be more effective to begin with a standard mesh and to embed it in an appropriate domain for use in the finite element system.

7.4.2 Implementing Remainder of System Outline

Future work is obviously required to test and implement the remaining items identified in this research as basic requirements for a computer-based freeform surface design system. As the current work includes a proof-of-concept system for only a subset of the requirements identified, this remains an area for future work.

7.4.3 Physical Interface Improvements

It has become very clear that the process of creating three-dimensional freeform surface forms is an extremely complex task. Automotive stylists and modelers simultaneously use their hands, wrists, elbows and legs to drag a sweep curve in order to modify a clay model. Clearly a simple mouse interface cannot hope to capture the breadth of control necessary to modify a freeform surface in a natural and intuitive way.

While this conclusion has become apparent, it remains to be determined how to address this shortcoming. Clearly, working in a fully or partially immersive virtual reality environment may go some of the way toward addressing certain aspects of this problem. However, a great deal more will be required in order to let the resident of the virtual environment interact and modify surfaces intuitively.

Some components of this additional technology may already exist. Clearly, however, many more do not. For instance, the process of creating tape (key-line) drawings as embodied in the Tape Tool could lend itself to an excellent, as yet non-existent, physical interface. Such an interface would involve the use of both hands. The right handed version of the tool would involve tracking the position of the left index finger in world space while also tracking the position of the right index finger as a relative displacement. Conceptually, this would be a very simple and effective interface for creating smooth curves in a free-hand way – something that is not possible in current systems.

7.4.4 Gestural Interface

Another avenue of future work and research is the development of more sophisticated versions of the Shape-From-Shading Tool. For instance, instead of interpreting the user's shading literally, a tool which interpreted a user's gestures intelligently would provide another means of extremely rapid surface modification.

By dragging the pointer across a surface, a designer would be able to modify the surface in some controllable way. For instance, if this tool were implemented in conjunction with a pressure sensitive pen and tablet, a user could use pressure to control some setting for the tool.

7.4.5 Parametric Modifications

An obviously desirable feature in a design system based upon the tools set out here would be a *parametric design* component. Thus, for each surface modification defined by the user, that modification can be defined and re-invoked with modified parameters at some later time.

Thus, on the path toward creating a final design, the designer creates an ordered sequence of surface modifications. If each of these modifications were completely defined, the user could conceivably return to any particular modification, edit it, and then re-execute the steps that followed. The ability to return to earlier modifications would facilitate the broad design experimentation often integral to the design process.

7.4.6 Beyond Existing Metaphors of Design

As alluded to earlier, the list of requirements set down in this research should be considered a minimal set of tools that would enable intuitive aesthetically-constrained freeform surface design in a computer environment. However, it would be foolish to assume that this is a complete listing of what could be implemented in this application domain.

The decision to base this list on what is currently a part of the freeform design process in the automotive styling field was based on the realization that it would be a complete – if minimal – set of tools. Once designers adopt computational tools in their work, they will surely develop new approaches to computational freeform surface design that are, as yet, unobserved.

Beyond additional approaches that will certainly be developed within automotive styling itself, the use of these tools by other design professionals will also enrich the tool-set over time. These other professionals will undoubtedly bring their own design experiences to bear on the problems of computational freeform surface design.

It will clearly be important to study and observe these additional approaches and to enhance the system with new tools that facilitate such alternative design approaches.

7.5 Final Word

Thus, while this work has identified and developed tools that better reflect the ways in which creative designers conceive of and develop their designs, a great deal clearly remains to be done. This research has aimed to be, and hopefully is, nothing more than a good start.

Bibliography

[AAB+87] E. Andersson, R. Andersson, M. Boman, B. Elmroth, T. Dahlberg, and B. Johansson. Automatic construction of surfaces with prescribed shape. *Computer-Aided Design*, 20(6):317–324, July/August 1987.

[Bar84] A. H. Barr. Global and local deformations of solid primitives. In *Computer Graphics (Proc. SIGGRAPH 84)*, pages 21–31, July 1984.

[Bat82] Klaus-Jürgen Bathe. *Finite Element Procedures in Engineering Analysis.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[BB83] B.A. Barsky and J.C. Beatty. Local control of bias and tension in Beta-splines. *ACM Transactions on Graphics 2*, pages 109–134, April 1983.

[BCIZ65] G.P. Bazeley, Y.K. Cheung, B.M. Irons, and O.C. Zienkiewicz. Triangular elements in plate-bending – conforming and non-conforming solutions. In *Proceedings of the Conference on Matrix Methods in Structural Mechanics.* Wright Patterson A.F.B., Ohio, October 1965.

[Bee86] Etienne Beeker. Smoothing of shapes designed with free-form surfaces. *Computer-Aided Design*, 18(4):224–232, May 1986.

[Bes87] Maurice Besset. *LeCorbusier: To Live with the Light.* Rizzoli International Publications, Inc., New York, 1987.

[Béz66] Pierre Bézier. Définition numérique des courbes. *Automatisme*, XI(12):625–632, 1966.

[Béz67] Pierre Bézier. Définition numérique des surfaces. *Automatisme*, XII(1):17–21, 1967.

[Béz68] Pierre Bézier. Procédé de définition numérique des courbes et surfaces non mathématiques. *Automatisme*, XII(5):189–196, 1968.

[Béz74] Pierre Bézier. Mathematical and practical possibilities of UNISURF. In Barnhill and Riesenfeld [BR74], pages 259–302.

[BFFH88] R.E. Barnhill, G. Farin, L. Fayard, and H. Hagen. Twists, curvatures and surface interrogation. *Computer-Aided Design*, 20(6):341–346, 1988.

[BFH86] J.M. Beck, R.T. Farouki, and J.K. Hinds. Surface analysis methods. *IEEE Computer Graphics and Applications*, 6(12):18–36, December 1986.

[BFK84] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. In R.E. Barnhill and W. Böhm, editors, *Computer-Aided Geometric Design*, volume 1. North-Holland, July 1984.

[BL99] Joan Bergós and Marc Llimargas. *Gaudí: The Man and his Work.* Bulfinch Press, Boston, 1999.

[Bon86] Luisa Bonfiglioli. An algorithm for silhouette of curved surfaces based on graphical relations. *Computer-Aided Design*, 18(2):95–101, March 1986.

[BP92] M. Bichsel and A. P. Pentland. A simple algorithm for shape from shading. In *IEEE Proceedings on Computer Vision and Pattern Recognition 1992*, pages 459–465, Champaign, Ill., USA, 1992.

[BR74] R.E. Barnhill and R.F. Riesenfeld, editors. *Computer Aided Geometric Design.* Academic Press Inc., New York, NY, 1974.

[Bun00] Carey Bunks. *Grokking the GIMP.* New Riders Publishing, 2000.

[BV89] S. Bedi and G.W. Vickers. Surface lofting and smoothing with skeletal-lines. *Computer-Aided Geometric Design 6*, pages 87–96, North Holland 1989.

[BW89a] M.I.G. Bloor and M.J. Wilson. Blend design as a boundary value problem. In Wolfgang Straßer and Hans-Peter Seidel, editors, *Theory and Practice of Geometric Modeling*, pages 221–234. Springer, 1989.

[BW89b] M.I.G. Bloor and M.J. Wilson. Generating blend surfaces using partial differential equations. *Computer-Aided Design*, 21(3):165–171, April 1989.

[BW89c] M.I.G. Bloor and M.J. Wilson. Generating n-sided patches with partial differential equations. In Barnshaw and Wyvill, editors, *New Advances in Computer Graphics, Proceedings of CG international '89*, pages 129–145. Springer, 1989.

[BW90a] M.I.G. Bloor and M.J. Wilson. Representing PDE surfaces in terms of B-splines. *Computer-Aided Design*, 22(6):324–331, 1990.

[BW90b] M.I.G. Bloor and M.J. Wilson. Using partial differential equations to generate free-form surfaces. *Computer-Aided Design*, March 1990.

[BW94] Malcolm I. G. Bloor and Michael J. Wilson. Interactive design using partial differential equations. In Nicholas S. Sapidis, editor, *Designing Fair Curves and Surfaces*, chapter 9, pages 231–251. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pennsylvania, 1994.

[BW95] M.I.G. Bloor and Hans Wilson, M.J. nd Hagen. The smoothing properties of variational schemes for surface design. *Computer Aided Geometric Design*, 12:381–394, 1995.

[Cas87] Malcolm Casale. Free-form solid modeling with trimmed surface patches. *IEEE Computer Graphics and Applications*, 7(1), January 1987.

[CBP97] Yifan Chen, Klaus-Peter Beier, and Dimitris Papageorgiou. Direct highlightline modification on NURBS surfaces. *Computer Aided Geometric Design*, 14:583–601, 1997.

[CC78] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, pages 350–355, November 1978.

[Cel90] George Celniker. *ShapeWright: Finite Element Based Free-Form Shape Design.* PhD thesis, MIT, 1990.

[CF84] Gen-zhe Chang and Yu-yu Feng. An improved condition for the convexity of Bernstein-Bézier surfaces over triangles. *Computer Aided Geometric Design*, 1, 1984.

[Cla76] J.H. Clark. Designing surfaces in 3D. *Communications of the ACM*, 19(8):454–460, August 1976.

[CM69] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *ACM National Conference*, pages 157–172, New York, 1969.

[Coo67] S.A. Coons. Surfaces for the computer-aided design of space forms. Technical report, M.I.T. MAC-TR-41, available from NTIS U.S. Department of Commerce, Springfield, VA 22151, 1967.

[Coq87] Sabine Coquillart. A control-point based sweeping technique. *IEEE Computer Graphics and Applications*, 7(11):36–45, November 1987.

[Coq90] Sabine Coquillart. Extended free-form deformation: A sculpting tool for 3D geometric modeling. In *Computer Graphics (Proc. SIGGRAPH 90)*, pages 187–194, August 1990.

[CT65] Ray W. Clough and James L. Tocher. Finite element stiffness matrices for analysis of plate bending. In *Proceedings of the Conference on Matrix Methods in Structural Mechanics.* Air Force Institute of Technology, Wright Patterson A.F.B., Ohio, October 1965.

[DD87] Dennis J. De Witt and Elizabeth R. De Witt. *Architecture of Europe: A Guide to Buildings Since the Industrial Revolution.* E. P. Dutton, New York, 1987.

[de 84] Ignasi de Solá-Morales. *Gaudí.* Ediciones Polígrafa, S. A., Barcelona, Spain, 1984.

[Dem93] James Demmel. *Numerical Linear Algebra.* Berkeley Mathematics Lecture Notes. Department of Mathematics and Computer Science Division, Berkeley, CA 94720, 1993.

[DeR85] Tony D. DeRose. Geometric continuity: a parametrization independent measure of continuity for computer aided geometric design. Technical Report UCB/CSD 86/255, Computer Sciences Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1985.

[DLG90] N. Dyn, D. Levin, and J. A Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[EV98] P. A. Elsas and J. S. M. Vergeest. Displacement feature modelling for conceptual design. *Computer-Aided Design*, 30(1):19–27, 1998.

[Fan94] Lian Fang. *Physically-Based Methods for Parametric Curve and Surface Reconstruction.* PhD thesis, Massachusetts Institute of Technology, 1994.

[Far82a] G. Farin. A construction for visual C^1 continuity of polynomial surface patches. *Computer Graphics and Image Processing*, 20:272–282, 1982.

[Far82b] G. Farin. Designing a C^1 surface consisting of triangular cubic patches. *Computer-Aided Design*, 14:253–256, 1982.

[Far85] Gerald Farin. A modified Clough-Tocher interpolant. *Computer Aided Geometric Design*, 2:19–27, 1985.

[Far86] G. Farin. Triangular Bernstein-Bézier patches. *Computer-Aided Geometric Design*, pages 83–127, North Holland 1986.

[Far90] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press Inc., Harcourt Brace Jovanovich Publishers, Boston, 3 edition, 1990.

[Fer64] J. Ferguson. Multivariable curve interpolation. *Journal of the ACM*, 2(2), 1964.

[FFJ88] D.R. Ferguson, P. Frank, and A. Jones. Surface shape control using constrained optimization on the B-spline representation. *Computer-Aided Geometric Design*, 5:87–104, 1988.

[FG95] Lian Fang and David C. Gossard. Multidimensional curve fitting to unorganized data points by non-linear minimization. *Computer-Aided Design*, 27(1):48–58, 1995.

[FS88] Gerald Farin and Nicholas Sapidis. Shape representation of sculpted objects: the fairness of curves and surfaces. In *Proceedings of the Sea Grant Conference.* MIT, October 1988.

[GC80] J.A. Gregory and P. Charrot. A C^1 triangular interpolation patch for computer-aided geometric design. *Computer Graphics and Image Processing*, 13:80–87, 1980.

[Geh95a] Frank O. Gehry. Guggenheim Museum in Bilbao: Frank O. Gehry. *ARCH+*, 128:24–25, September 1995.

[Geh95b] Frank O. Gehry. Walt Disney Concert Hall: Frank O. Gehry. *ARCH+*, 128:20–23, September 1995.

[Geo93] Priamos N. Georgiades. Using graphs of bivariate functions to locally represent and modify surfaces. *Computer Aided Geometric Design*, 10:453–463, 1993.

[GG92] Priamos N. Georgiades and Donald P. Greenberg. Locally manipulating the geometry of curved surfaces. *IEEE Computer Graphics and Applications*, pages 54–64, January 1992.

[GL98] S. Guillet and J. C. Léon. Parametrically deformed free-form surfaces as part of a variational model. *Computer-Aided Design*, 30:621–630, 1998.

[GLW96] Günther Greiner, Joachim Loos, and Wieger Wesselink. Data dependent thin plate energy and its use in interactive surface modeling. In J. Rossignac and F. Sillion, editors, *Eurograhics '96*, volume 15, pages 175–185. Eurographics Association, 1996.

[GR74] W.J. Gordon and R.F. Riesenfeld. B-spline curves and surfaces. In Barnhill and Riesenfeld [BR74], pages 95–126.

[Gre74] J.A. Gregory. Smooth interpolation without twist constraints. In R.E. Barnhill and R.F. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 71–87. Academic Press Inc., New York, NY, 1974.

[Gre85] J.A. Gregory. Interpolation to boundary data on the simplex. *Computer-Aided Geometric Design 2*, pages 43–52, North Holland 1985.

[Gri00] Larry Gritz. Blue Moon Rendering Tools Homepage. http://www.bmrt.org, 2000.

[GW72] William J. Gordon and James A. Wixom. Pseudo-harmonic interpolation on convex domains. Technical Report GMR-1248, Research Laboratories, General Motors Corporation, Warren, Michigan, August 1972.

[GWPS76] N. E. Gibbs, G William, J. Poole, and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile for a sparse matrix. *SIAM J. of Numerical Analysis*, 13(2):236–250, 1976.

[HB89] Berthold Klaus Paul Horn and Michael J. Brooks. *Shape From Shading.* MIT Electrical Engineering and Computer Science Series. MIT Press, 1989.

[Her85] Gary Herron. Smooth closed surfaces with discrete triangular interpolants. *Computer Aided Geometric Design*, 2:297–306, 1985.

[HH96] Dave Hutchinson and Terry Hewitt. Rapidly visualizing isophotes. *Journal of Graphics Tools*, 1(3):7–12, 1996.

[HHK92] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form surfaces. In *Computer Graphics (Proc. SIGGRAPH 92)*, pages 177–184, July 1992.

[Hor86] Berthold Klaus Paul Horn. *Robot Vision.* MIT Electrical Engineering and Computer Science Series. MIT Press, 1986.

[Hos84] Josef Hoschek. Detecting regions with undesirable curvature. *Computer Aided Geometric Design*, 2(1):183–192, 1984.

[Hos85] Josef Hoschek. Smoothing of curves and surfaces. *Computer Aided Geometric Design*, 2:97–105, 1985.

[Hos98] Josef Hoschek. Approximation of surfaces of revolution by developable surfaces. *Computer-Aided Design*, 30(10), 1998.

[HS87] H. Hagen and G. Schulze. Automatic smoothing with geometric surface patches. *Computer-Aided Geometric Design*, 4:231–236, 1987.

[Jon88] A.K. Jones. Nonrectangular surface patches with curvature continuity. *Computer-Aided Design*, 20(6):325–335, 1988.

[JPW91] T. W. Jensen, C. S. Petersen, and M. A. Watkins. Practical curves and surfaces for a geometric modeler. *Computer Aided Geometric Design*, 8(5):357–370, November 1991.

[Kas96] Praveen Kashyap. Improving Clough-Tocher interpolants. *Computer Aided Geometric Design*, 13:629–651, 1996.

[Kje83a] J.A. Kjellander. Smoothing of bicubic parametric surfaces. *Computer-Aided Design*, 15(5):288–293, 1983.

[Kje83b] J.A. Kjellander. Smoothing of cubic parametric splines. *Computer-Aided Design*, 15(3):175–179, May 1983.

[KK87] E. Kaufmann and R. Klass. Smoothing surfaces using reflection lines for families of splines. *Computer-Aided Design*, 20(6):312–316, July/August 1987.

[KR90] M. Kallay and B. Ravani. Optimal twist vectors as a tool for interpolation a network of curves with a minimal energy surface. *Computer Aided Geometric Design*, 7, 1990.

[Kre91] Erwin Kreyszig. *Differential Geometry.* Dover, New York, 1991.

[Kuo98] M H Kuo. Reconstruction of quadric surface solids from three-view engineering drawings. *Computer-Aided Design*, 30(7):517–527, 1998.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snake: Active contour models. *International Journal of Computer Vision*, 3(1), 1988.

[LeC67] LeCorbusier. *Gaudí.* Ediciones Polígrafa, Barcelona, Spain, 1967.

[LeC95] Annette LeCuyer. Entwerfen am Computer: Frank Gehry und Peter Eisenmann. *ARCH+*, 128:26–29, September 1995.

[Lee90] E.T.Y. Lee. Energy, fairness and a counter example. *Computer-Aided Design*, 22(1):37–40, January/February 1990.

[LGS99] Joachim Loos, Günther Greiner, and Hans-Peter Seidel. Modeling of surfaces with fair reflection line pattern. In *Shape Modeling International '99.* IEEE, 1999.

[Liu95] Xiong-Wei Liu. Five-axis NC cylindrical milling of sculptured surfaces. *Computer-Aided Design*, 27(12):887–894, 1995.

[LM95] Richard C. Levene and Fernando Márquez Cecilia, editors. *El Croquis – Frank Gehry, 1991-1995.* El Croquis, S.L., 1995.

[Loe79] Raymond Loewy. *Industrial Design.* Fournier Artas Grafica, S. A., Spain, 1979.

[Loo94] Charles Loop. Smooth spline surfaces over irregular meshes. In *ACM SIGGRAPH 94 Conference Proceedings*, pages 303–310. Addison-Wesley, 1994.

[LP88] N.J. Lott and D.I. Pullin. Methods for fairing B-spline surfaces. *Computer-Aided Design*, 20(10):597–604, December 1988.

[LP98] Stefan Leopoldseder and Helmut Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, 30(7), 1998.

[LPV95] A. F. Lennings, J. C. Peters, and J. S. M. Vergeest. An efficient integration of algorithms to evaluate the quality of freeform surfaces. *Computers & Graphics*, 19(6):861–872, 1995.

[Lyn99] Greg Lynn. Greg Lynn develops a system for generating biomorphic manufactured houses. *Architectural Record*, December 1999.

[LZ95] Xiaochun Liu and Yun Zhu. A characterization of certain C^2 discrete triangular interpolants. *Computer Aided Geometric Design*, 12:329–348, 1995.

[Mac88] B.L. MacCarthy. Quintic splines for kinematic design. *Computer-Aided Design*, 20(7):406–415, September 1988.

[Mae95] Takashi Maekawa. Lecture notes on differential geometry. MIT Course 13.472J, 2.158J, 1.128J, February 1995.

[Man99] Stephen Mann. Cubic precision Clough-Tocher interpolation. *Computer Aided Geometric Design*, 16:85–88, 1999.

[MN00] T. Murakami and N. Nakajima. DO–IT: deformable object as input tool for 3-D geometric operation. *Computer-Aided Design*, 32:5–16, 2000.

[MQV00] C. Mandal, H. Qin, and B. C. Vemuri. A novel FEM-based dynamic framework for subdivision surfaces. *Computer-Aided Design*, 32:479–497, 2000.

[MS92] Henry P. Moreton and Carlo H. Séquin. Functional optimization of fair surface design. In *Computer Graphics (Proc. SIGGRAPH 92)*, pages 167–176, July 1992.

[Mun87] F. Munchmeyer. Shape interrogation: A case study. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 291–302. SIAM, 1987.

[MWP96] T. Maekawa, F. E. Wolter, and N. M. Patrikalakis. Umbilic and lines of curvature for shape interrogation. *Computer Aided Geometric Design*, 13:133–161, 1996.

[NG00] J. Cotrina Navau and N. Pla Garcia. Modelling surfaces from planar irregular meshes. *Computer Aided Geometric Design*, 17:1–15, 2000.

[Nie74] G.M. Nielson. Some piecewise polynomial alternatives to splines in tension. In R.E. Barnhill and R.F. Riesenfeld, editors, *Computer-Aided Geometric Design.* Academic Press, 1974.

[NR83] H. Nowacki and D. Reese. Design and fairing of ship surfaces. In R.E. Barnhill and W. Böhm, editors, *Surfaces in CAGD*, pages 121–134. North-Holland, Amsterdam, 1983.

[O'R94] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, Cambridge, UK, 1994.

[Pen86] Alex P. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:292–331, 1986.

[Pet95] Jörg Peters. Biquartic C^1-surface splines over irregular meshes. *Computer-Aided Design*, 27(12):895–903, 1995.

[Pet96] Jörg Peters. Curvature continuous spline surfaces of irregular meshes. *Computer Aided Geometric Design*, 13:101–131, 1996.

[Poe84] T. Poeschl. Detecting surface irregularities using isophotes. *Computer Aided Geometric Design*, 1(2):163–168, 1984.

[Pra78] A. Pramila. Ship hull surface design using finite elements. *International Shipbuilding Progress*, 25(284):97–109, 1978.

[PT87] L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. *Computer-Aided Design*, 19(9):485–498, 1987.

[PT97] L. Piegl and W. Tiller. *The NURBS Book.* Springer Verlag, 1997.

[PW99] Helmut Pottmann and Johannes Wallner. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design*, 16:539–556, 1999.

[RE00] A. Raviv and G. Elber. Three-dimensional freeform sculpting via zero sets of scalar trivariate functions. *Computer-Aided Design*, 32:513–526, 2000.

[Ree83] W. T. Reeves. Particle systems - a technique for modelling a class of fuzzy objects. *Computer Graphics*, 17(3):359–376, 1983.

[RS93] Thomas Reuding and Milan Sreckovic. Automatic local updating of cad surface models. In *SIAM Conference on Geometric Design.* SIAM, 1993.

[RSR83] David F. Rogers, G. Satterfield, and Francisco A. Rodriguez. Ship hulls, B-splines surfaces, and cad/cam. *IEEE Computer Graphics and Applications*, 3(9), December 1983.

[SB89] D.J.T. Storry and A.A. Ball. Design of an n-sided surface patch from hermite boundary data. *Computer Aided Geometric Design*, 6:111–120, 1989.

[Sch66] D.G. Schweikert. An interpolation curve using a spline in tension. *Journal of Math and Physics*, 45:312–317, 1966.

[Sch93] Daniel L. Schodek. *Structure in Sculpture*, chapter 15, page 200. MIT Press, Cambridge, MA, 1993.

[SF90] N. Sapidis and G. Farin. Automatic fairing algorithm for B-spline curves. *Computer-Aided Design*, 22(2):121–129, March 1990.

[SF96] Meng Sun and Eugene Fiume. A technique for constructing developable surfaces. In *Graphics Interface*, pages 176–185, May 1996.

[Smy93] Evan Smyth. Computer-aided interactive sculpting. Paper describing implementation for MIT course, December 1993.

[SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Computer Graphics (Proc. SIGGRAPH 86)*, pages 151–160, August 1986.

[SS97] Byeong-Seok Shin and Yeong Gil Shin. Fast 3D solid model reconstruction from orthographic views. *Computer-Aided Design*, 30(1):63–73, 1997.

[Ste81] K.A. Stevens. The visual interpretation of surface contours. *Artificial Intelligence*, 17:47–75, 1981.

[Str88] Dirk J. Struik. *Lectures on Classical Differential Geometry.* Dover, New York, 1988.

[TF97] Holger Theisel and Gerald Farin. The curvature of characteristic curves on surfaces. *IEEE Computer Graphics and Algorithms*, November/December 1997.

[Til83] Wayne Tiller. Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, September 1983.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastic deformable models. *ACM SIGGRAPH 87 Conference Proceedings*, 21(4), July 1987.

[TQ94] Demetri Terzopoulos and Hong Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[TWK87] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1987.

[UBW99] Hassan Ugail, Malcolm I.G. Bloor, and Michael J. Wilson. Techniques for interactive design using the PDE method. *ACM Transactions on Graphics*, 18(2):195–212, 1999.

[Vas97] Tzvetomir Ivanov Vassilev. Interactive sculpting with deformable nonuniform B-splines. *Computer Graphics Forum*, 16(4):191–199, 1997.

[vO96] C.W.A.M. van Overveld. Painting gradients: free-form surface design using shading patterns. In *Graphics Interface*, pages 151–158, May 1996.

[Whe86] T. Whelan. A representation of a C^2 interpolant over triangles. *Computer Aided Geometric Design*, 3:53–66, 1986.

[Wic87] Hans Wichmann. *Design Process Auto.* Die Neue Sammlung Staatliches Museum fuer angewandte Kunst, Prinzregentenstrasse 3, Muenchen, Germany, 1987.

[Woo88] C.D. Woodward. Skinning techniques for interactive B-spline surface interpolation. *Computer-Aided Design*, 20(8), October 1988.

[WW94] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *ACM SIGGRAPH 94 Conference Proceedings*, pages 247–256, July 1994.

[Yam93] Yasusato Yamada. Clay modeling: Techniques for giving three-dimensional from to idea. *Car Styling: Extra Issue*, 93 1/2, July 1993.

[YPM00] Guoxin Yu, Nicholas M. Patrikalakis, and Takashi Maekawa. Optimal development of doubly curved surfaces. *Computer Aided Geometric Design*, 17:545–577, 2000.

[ZB97] J. J. Zheng and A. A. Ball. Control point surfaces over non-four-sided areas. *Computer Aided Geometric Design*, 14:807–821, 1997.

[ZC98] Caiming Zhang and Fuhua (Frank) Cheng. Removing local irregularities of NURBS surfaces by modifying highlight lines. *Computer-Aided Design*, 30(12):923–930, 1998.

[Žen72] A. Ženišek. Hermite interpolation on simplices in the finite element method. In *Proceedings of Equadiff. III, 3rd Czech. Conf. Diff. Equa. Appl.*, Brno, 1972.

[Zer85] Rainer Zerbst. *Antoni Gaudí.* Taschen Verlag GmbH, Köln, Germany, 1985.

[ZT89] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method.* McGraw Hill, England, 1989.