REPRESENTATION OF AND MODELING WITH ARBITRARY DISCONTINUITY CURVES IN SCULPTURED SURFACES

by

Marc S. Ellens

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

The University of Utah

August 1997

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Marc S. Ellens

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Elaine Cohen

Richard F. Riesenfeld

Jamie Painter

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Marc S. Ellens in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Elaine Cohen
Chair, Supervisory Committee

Approved for the Major Department

Tom Henderson
Chair/Dean

Approved for the Graduate Council

Ann W. Hart
Dean of The Graduate School

ABSTRACT

Representing data after it has undergone a fundamental topological change, such as cracking, ripping, or folding, or after the introduction of arbitrary feature curves, as happens during the creation of darts, corners, or fractures, continues to be a significant challenge. Ideally, the representation is modified without having to reformulate the representation entirely. If the original model is composed of faceted polyhedra, it is possible to do this. However, many models today are being represented by smooth parametric tensor product surfaces such as B-splines, which do not easily support arbitrary discontinuities. During the design process, when discontinuities are introduced, such models are often tesselated into triangles, which would henceforth be the model's representation. In this case, the resulting model is often not useful for further design. This thesis introduces an extension of the B-spline surface representation, called the *torn B-spline surface*. The torn B-spline representation provides flexibility not previously found in similar parametric surfaces by incorporating *tear* curves, *crease* curves, and other arbitrary $C^{(-1)}$ feature curves into the representation itself. Simulation events or other design processes which result in discontinuities in the representation do not necessitate a change in representation, and it is possible to use B-spline design methods on the resulting torn surface model. This makes design with discontinuities more viable. The representation and associated algorithms used to support it are introduced, as well as some higher-order design operators which take advantage of this representation and some example applications.

To Julie, Karys, and Serena.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

This work was made possible by the advice and support of a great number of people. Thanks go to: my wife and family for putting up with the long hours and lack of attention (I am going to be home more now); Elaine Cohen and the rest of my committee, for excellent technical advice; my coworkers, for putting up with my constant jabber; my fellow graduate students, for not letting me give up and providing someone to compete with ("It's not my fault, really!"); Beth Cobb, for her clear-thinking advice and friendship; my friends at Mountain Springs, for pushing me to finish this without knowing what it was I was doing—and loving me anyway; my parents, for being my parents; and God, for being my God. May I never have to do this again.

This work was supported in part by DARPA (N00014-92-J-4113) and the NSF and DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219). All opinions, findings, conclusions, or recommendations expressed in this document are mine and do not necessarily reflect the views of the sponsoring agencies.

CHAPTER 1

INTRODUCTION

In the field of computer-aided geometric design (CAGD), use of an appropriate representation is the key to effectively conveying a structure's geometry. There are many tradeoffs among the representations, based on factors such as whether or not the data accurately represents the original model, which will be referred to as *fidelity*, and the number of common operations under which the set of representable models is closed which measures the *completeness* of the representation. Other factors to be weighed include ease of use, size, speed, and flexibility. The flexibility of a representation is measured in terms of the number of its supporting operations in the same way a mathematician may evaluate an algebra in terms of the number of common operators that can be applied. Consider the tradeoffs between polygonal representations, various parametric tensor product surface representations, and constructive solid geometric (CSG) representations. In the ideal world, all shapes would be represented exactly and the particular representation would not be an issue. However, the real world usually has more complexity than we are able to represent, so we approximate. As a rule, the more complexity a model has, the more it is approximated. The triangle or otherwise faceted representation is flexible and closed under most operations at the expense of larger size and decreased fidelity. The tensor product B-spline representation is compact and easy to manage at the expense of being slower and more incapable of representing the results of some operations. A CSG representation is even more compact and easier to understand and use while being even less flexible. However, technological advances in computer speed and memory have reduced the impact of the size and speed requirements of a representation. The crucial factors have become fidelity, completeness, flexibility, and ease of use.

The driving force behind this research is the need to provide a representation for which operations that introduce discontinuities into tensor parametric surfaces are closed. Operations in which discontinuities are introduced as a result of a simulation or other automatic process are particularly difficult since the designer often has little control over the outcome and the results are not easily representable by current parametric tensor product surface representations. Consider the following examples where parametric tensor product surfaces are likely to be used.

The first example is the much studied area of thin plate deformations[74]. Thin plate deformations are used for modeling the behavior of everything from cloth[81] to steel. Thin plate dynamics can be modeled with simple springs and dashpots in a linear constraint/feedback system or in a complex nonlinear finite element optimization system. In either case the physical characteristics of the plate may cause the plate to tear or fracture under stress, causing a change that is probably not supported by the model's representation. The simulation results may be translated into a different representation which may be capable of representing the discontinuity but fails to retain the smoothness information within the rest of the model. Sometimes the model is reconstructed with explicit constraints holding together the new edges with the old smoothness information. All cases result in additional work for the designer, particularly if the results will be used in the context of further design operations or analysis. Ultimately, information about the model is lost during this process.

Another related example is *stamping*[27, 3, 36]. In this process, a thin malleable material is forced under pressure to assume a particular shape by compressing the material between two forms. The simulation of this process is extremely difficult since there is high pressure and heat, both of which may alter the state and the dynamics of the original material. Fractures, tears, and creases are common and cause difficulty in both the simulation and the representation of the model itself.

In the field of geology we see another example of physical simulation resulting in unrepresentable complex shapes. Here the discontinuities are three-dimensional as earthquakes and other natural forces cause rock layers to fracture and slide past one another creating complex systems of slips and folds. An initial representation may be parametric tensor product surfaces that form the boundaries of the rock layers, stacked on each other to make a *nonmanifold* three-dimensional model. Once the rock layers separate, indication of the three-dimensional nature of the crack is virtually impossible with current parametric tensor product representations.

In the medical field, physicians can simulate procedures such as plastic and reconstructive surgery[62] allowing them to make better planning decisions. Current representations in this area are polygonal although actual skin is rarely faceted. The ability to support arbitrary continuity features in parametric boundary representation models could be well used by this field.

Finally, a designer using a CAD system may want to include continuity features within a surface, such as creases or tears. These features may drive a particular functional aspect, such as aerodynamics, or an aesthetic aspect of the design.

Currently, triangles are the most common representation in these cases, because they are flexible and easy to use and they support the arbitrary topologies and continuity features which may result. However, design with parametric surfaces, and in particular, tensor product B-splines, is becoming more prevalent. Current design techniques need to provide adequate support for these surfaces and the by-products of their design processes. Ideally, a design operation that introduces discontinuities should result in models that are members of the original representational set. Currently this is not the case.

In these cases, flexibility and fidelity appear to be the two most critical factors. To produce an accurate model, the initial representation must be accurate and all prior operations must retain that information as well as reliably incorporate new information during the process. Of particular concern are the continuity features contained in the representation. Flexibility is the key to obtaining an accurate initial representation, whereas fidelity is the key to retaining the accuracy through modifications. The faceted representations usually consist of many small facets that together approximate the continuity information present in the large model.

Usually continuity information is maintained by an equation which measures the energy present between adjacent facets and so determines a smoothness measure of the model. Continuity features in the larger model are usually identified along the boundaries between facets and the energy equation is suitably modified to reflect the change in continuity. Higher-order tensor product surfaces provide a more accurate initial model but are unable to represent all the continuity features which may be introduced. A *feature curve* cannot be introduced into a tensor product surface without significantly altering the representation. This, in many cases, prevents the use of some available design techniques since this type of smoothness information cannot be represented by a single tensor product surface.

The *torn B-spline surface* representation, initially presented in [29], is designed to bridge the gap between the faceted representations and the higher-order parametric representations. It provides the geometric flexibility of the faceted representations while providing the fidelity and size of the parametric representations. The key elements of the torn B-spline representation are the arbitrary $C^{(-1)}$ feature curves known as *tear curves*. The basis for this representation, the representation itself, and several of the more common evaluation routines applicable to this class of surfaces will be introduced, as well as some higher-order design operators which demonstrate the flexibility of the representation within a design system. Finally this representation will be applied to the problems in the examples introduced, demonstrating the effectiveness of the representation.

CHAPTER 2

BACKGROUND

The foremost consideration when assessing a representational need is determining the best representation class for the job in terms of flexibility and fidelity.

2.1 Representation Classes

Within a representation class, a given representation has the power to represent a particular set of models. The representation usually consists of a basic element, such as a triangle or surface, and its set of representable models can be classified by whether or not collections of these basic elements are used. In addition, representations vary in how closely they can approximate a given object, providing further classification.

A given representation also supports design operations, called *methods*, which are particular to that representation. For example, refinement is usually identified with parametric surfaces, in particular, those which have basis functions, such as tensor product B-splines. Other representations within this same class attempt to provide similar operations. The resulting models of these representation-linked operations generally stay within the same representation. Thus the set of representable models is closed with respect to these operations.

In addition to being used with their methods, representations can be used in a variety of design processes (often called operations as well) which can be modified to support a variety of representations, even those in other classes. For example, the tensor product B-spline and triangular-faceted representations can both be used in a physically based modeling process. The results of this set of operations may fall outside of the original representation's set of representable models, sometimes even outside the set of representable models for the class. Therefore, the representation or the representational class may not be closed with respect to these operations.

During the course of a design session, a designer constructs a model, modifies the model, possibly simulates some aspect of the model's function using the model itself, or makes modifications to the model through some automatic process. Then he may make changes to the model in response to feedback or examination of the resulting model's structure; which starts the process over again. If, at some point, the representation changes as a result of a particular operation, the feedback loop is altered. Design operations, in particular, methods, previously used for construction and modification may no longer be available and the designer may be incapable of making the necessary modifications without starting over. This is particularly true when the resulting model contains characteristics necessary for the final model.

Changing the representation class of the model during the modeling process can cause significant problems and force designers into using more cumbersome representations, dealing with less accuracy, choosing less intuitive design procedures or ultimately settling for less than what is required by the design specifications.

2.2 To Facet or Not to Facet...?

Within a complex design system, the choice of an internal representation affects both the interaction that the designer has with the system and the final outcome of the design process. If the choice were simple, there would not be much difference among design systems and the opinions that gave rise to them. As it is, design systems range from very low order, such as points with adjacency information (a.k.a. facets), to very high order, such as implicit surfaces[2]. Since many real-life objects have smooth, sculptured shapes, the challenge with the low-order faceted representation is to make the models look and behave like higher-order models without the size of the model becoming prohibitive[71, 46, 47]. The challenge with higher-order representations is to make them easy to use in a practical design system.

One of the primary reasons faceted representations are used within a design system is that they are well understood and well supported. Higher-order representations can be difficult to construct, manipulate, and simulate and are only recently being found in the larger commercial design systems. In contrast, the finite element method, the most popular simulation technique, caters to faceted representations (in two-dimensional simulations) since the meshes consist of interconnected data points (although higher-order physical relationships between mesh points are often used). Many design systems use tessellated models for this reason alone.

However low-order representations have disadvantages. Along with the size, which causes these representations to be difficult to manage and manipulate, another disadvantage is appearance. Faceted models have angular silhouettes over curved portions of their surfaces and can be subject to unwanted mach banding at adjacent edges[6]. In addition, higher-order representations are being actively investigated and are increasingly being used in commercial design systems. The computation time often consumed with higher-order representations is being countered with faster computers, making these representations a viable alternative to the traditional approach of using faceted models in a design system. Despite these disadvantages, facets are still popular and are implemented in systems which support a large range of design capabilities.

2.3 Tensor Product B-splines

Since tensor product B-splines have become commonplace in major design systems, it is important to understand the capabilities of the representation and have a clear idea of the extensions that can and need to be made to support the desired design operations. This thesis introduces the torn tensor product B-spline, a tensor product B-spline surface representation for which the set of representable models is closed under most design and simulation operations which may introduce discontinuities. This closure problem is difficult because the tensor product B-spline surface representation does not support discontinuities of arbitrary geometry. Since tensor product B-spline surfaces are being used more extensively in design processes, alternate representations which satisfy this closure requirement are more urgently needed. The problem is further characterized by looking closer at the examples mentioned earlier and the situations in which the need for representations closed under these operations arises and causes difficulty.

2.4 Physically Based Modeling

Several of these examples fall in the category of physically based modeling. Unfortunately, physically based modeling is a very general term which can be applied to just about any design or simulation environment which uses physical characteristics to help define the model. Even the use of lighting models such as radiant illumination (radiosity) could be considered physically based modeling because the resulting image is a product of a (albeit simplified) physical simulation. More commonly, however, physically based modeling refers to the use of physical characteristics to determine a model's shape, position and/or orientation in space. Material characteristics like mass, density, moments of inertia, elasticity, and plasticity combined with physical behaviors in context such as gravity, collision detection, and connectivity are examples of the physical characteristics considered in these design operations. A subclass of these problems deal with the physical characteristics of a single model. Multiple models require additional linkage and kinematic information. This thesis primarily addresses the single model case. The model may be a solid model whose basic elements have volume, or a boundary representation whose basic elements are surfaces and connected by constraints at the edges. Models discussed in this thesis are composed of parametric surfaces. Specifically, this thesis addresses manifold and nonmanifold boundary representations.

2.4.1 Thin Plate

The simplest and most well-understood dynamic simulation is that of the thin plate under tension[74, 39]. The thin plate problem is kept simple by assuming that the thickness of the plate does not contribute anything significant to the formulation of the problem. This allows a two-dimensional simulation of the plate which substantially reduces the complexity and the solution time. For the most part, standard thin plate dynamics is uninteresting since most simulations use a variation of the standard dynamics formulation. However, the use of continuous surface representations for thin plates can be difficult if the dynamics can support discontinuities such as tears or fractures. Although they may seem avoidable, these situations may occur in any instance in which plasticity is used. A case could be made that a model of dynamics without the capacity for fracture is not an accurate model, since it more accurately reflects real life.

Plasticity in materials engineering is most often represented by a stress-strain graph. The stress is the amount of force applied to a material, and the strain is the amount of deformation caused by the force. Although simplified greatly, the graph in Figure 2.1 is useful for reference[81]. Normally, a material's stress-strain graph is nonlinear and changes according to the history of the stress on the material. The elastic limit point (E) is the point up to which removal of the forces will cause the material to return to its original rest state. Stress beyond the elastic limit will cause the material to permanently deform. The material breaks when stressed beyond the breaking point (B).

Most parametric surface representations break down when the material reaches the breaking point. Either the discontinuity is ignored or a secondary representation is used, such as a triangular or other faceted tessellation or the visual representation

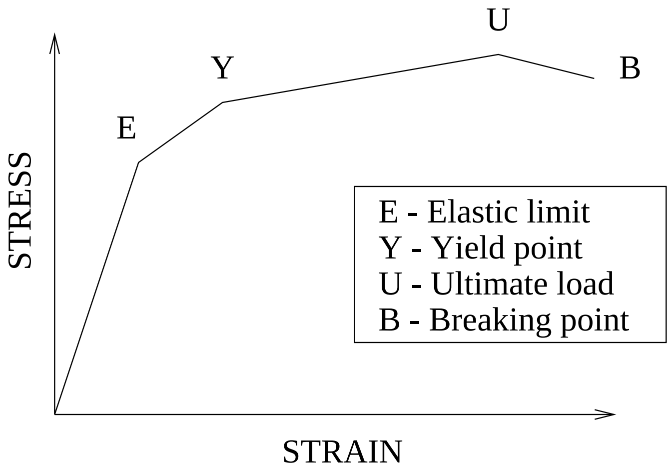

Figure 2.1. Stress-strain graph.

of the physical elements (e.g., plastic springs between control points[81]). In either case, the set of representable models is no longer closed under this simulation operation and the resulting representation may not retain smoothness information which is critical to the model.

2.4.2 Stamping

A more complex situation is the dynamics of stamping. In this manufacturing process, a thin sheet of material called a *blank* is cut to a specified shape and loaded into a press. The material is then deformed by forcing the material into a die by means of a punch. Although thin-plate dynamics play a large roll in the simulation of this process, a better simulation requires the use of material thickness and viscosity. The four general categories of critical problems in sheet metal forming are fracture, wrinkling or buckling, undesired sheet deformation, and springback[43, 36]. In some cases, however, these "problems" are not really problems but desired features. In the case of lance or emboss punching, (see Figures 2.2 and 2.3), the resulting fracture and the accompanying buckle are part of the design[27]. These simulation results need to be represented as accurately as possible. Although simulation of stamping processes are traditionally done using FEM, designing with higher-order parametric surfaces is more common, and the need for a consistent representation throughout the design process is becoming more evident.

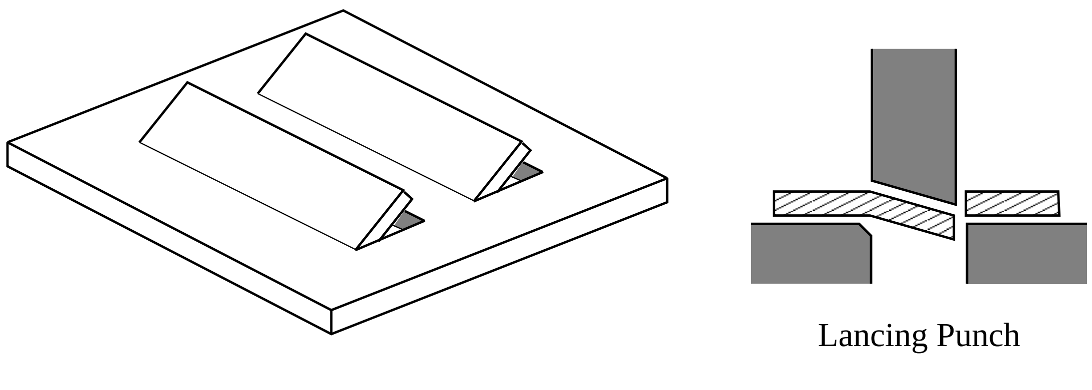

Figure 2.2. Lance forms.

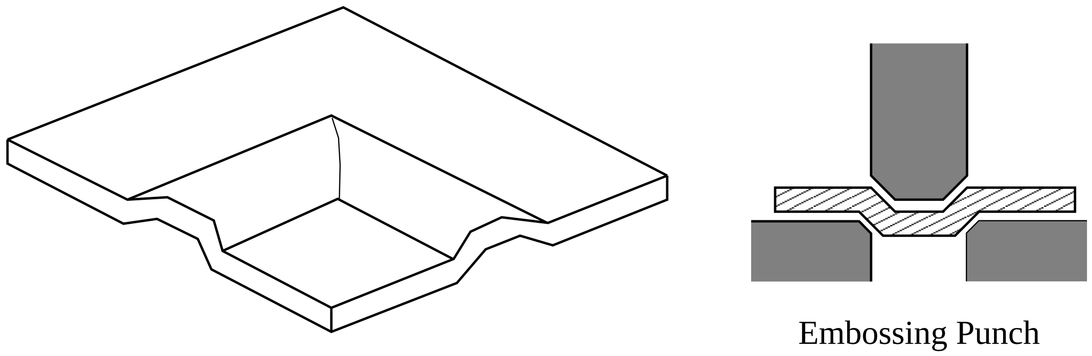

Figure 2.3. Embossing.

2.4.3 Geology

The study of fractures in rock layers is extremely complex and the representations used in the simulation of these interactions are inadequate at best. A good example of the need for discontinuities within a representation comes from the interaction of layers when a lower layer fractures and an upper layer deforms, or folds over the fracture (see Figure 2.4). This is known as *drape-folding*[89]. The different compositions of the layers may even cause the layers to separate resulting in a pocket between the layers. Other fractures may extend through the layers requiring that the representation track the fracture propagation through the different materials. The process of shearing also can create fractures and creases in the layers of material which are deformed[66]. Currently most simulation of geological phenomena is done with faceted representations using a constraint system solved by the finite element method. This is because of the frequent use of the finite element method in engineering disciplines and the current difficulty of representing fractures and other discontinuities in higher-order representations.

2.4.4 Medicine

Recently, Peiper has created some interest in surgical simulation, specifically plastic surgery[62, and references therein]. He primarily uses a triangular finite element mesh to represent the surface tissue during a finite element simulation. The ability to represent discontinuities in higher-order surfaces could have a serious

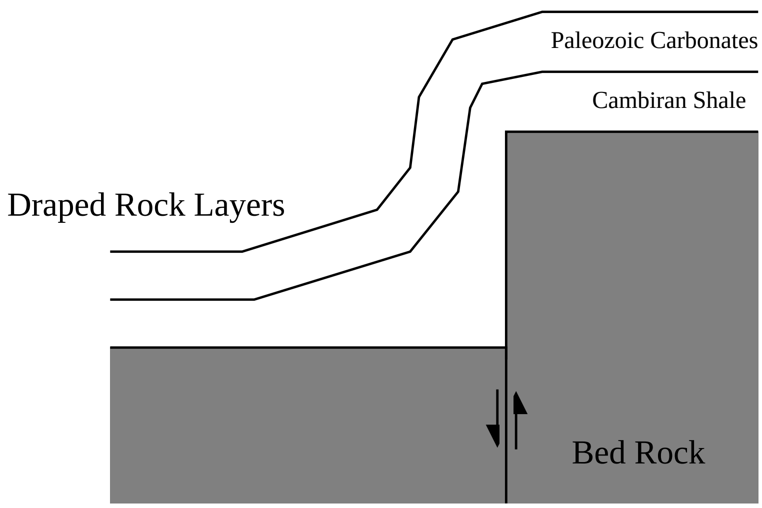

Figure 2.4. Drape folding.

impact on the planning and performance of various plastic surgery techniques. Currently most facial models are represented by polygons[84, 87]. However, parametric surfaces, and tensor product B-splines in particular, would be ideal candidates for representing skin, especially facial tissue, since smooth faces are more visually realistic for simulation. Although smoothness is generally not required for traditional animation[84], the addition of discontinuities to these smooth representations would provide a more realistic result for simulation of plastic and reconstructive surgery.

2.5 Design Techniques

The design processes in the above examples all employ similar techniques which are reviewed in this section. The most common techniques are methods for optimization of functions within the context of solving a system of constraints.

2.5.1 Constraints

There are a variety of ways to solve constraints depending on the characteristics of the constraint functions. For example, systems of linear constraints can be solved by singular value decomposition or QR factorization[72]. Most systems of constraint equations used for interactive physically based modeling are linear approximations to nonlinear systems[21, 35, 34] due to the speed and ease of solving these types of equations. If the process of solving the constraint is itself interesting, as it often is in the case of animation, there are other methods which can be used to iteratively obtain a solution in which the intermediate values are interesting. These constraint solving methods fall into three categories: 1) optimization, 2) dynamic constraints, and 3) reaction constraints[65].

2.5.2 Optimization

The most common of the optimization methods are penalty methods and Lagrangian constraints[64]. Other methods include finite differencing, simulate annealing, augmented Lagrangian constraints, and a host of other derivations of these methods[67]. It is these methods paired with a finite element mesh which comprise the finite element method (FEM)[10]. One difficulty of general optimization techniques is that they may require the computation of the gradient of the function being minimized or maximized and often these derivatives do not exist in closed form. Another difficulty is that most iterative optimization techniques are susceptible to getting caught in local minima or maxima, ultimately failing to find the globally optimal solution. In addition, intermediate values may not have any physical meaning, which may be a criteria if the optimization is being carried out in the context of an animation.

2.5.3 Dynamic Constraints

Dynamic constraints[7, 49, 4, 25] are systems of constraints which are solved by applying critically damped forces which are computed by inverse dynamics. The forces are thought of as occurring though time and therefore result in interesting intermediate solutions. Unfortunately, these techniques are often difficult to use because of their nonlinearity and large number of variables. Dynamic constraints are most often used in systems involving elasticity where the equations can be simplified[81, 88, 32](see Section 2.5.5).

2.5.4 Reaction Constraints

Another area of force-based constraint solving are reaction constraints. Additional forces are added to the system at the appropriate points in order to prevent the constraints from being violated. Typical uses of reaction constraints include path following and interpenetration prevention[5, 57]. The principal advantage to reaction constraints are that they are simple to compute. However, they often do not bear up well when applied to more complex problems[65].

2.5.5 Model Deformation

The animation industry has fueled interest in the theory of model deformation, with most of the theory developed from a physical basis. Elasticity and plasticity derived from the thin-plate model have been used in many instances to provide realistic motion and deformation. The standard methods minimize a variational derivation of an energy functional (similar to the thin-plate energy functional) over the model as a whole. The majority of the earlier work uses triangles to represent the final model[50, 77, 80, 76, 75, 78, 37, 38, 91, 20, 86, 15, 85]. It is only recently that these methods have been applied to splines in the works of Bloor and Wilson[13, 14], Welch and Witkin[90], Moreton and Sèquin[58], Celniker and Welch[21] and Terzopoulos and Qin[79, 68]. A unique method for deforming models based on vibration modes was developed by Pentland[63]. This "modal" dynamics method is only applicable to models described by closed-form functions.

2.6 Finite Elements

The finite element method is easily the most widely used simulation technique today[10, 48], and finite elements are used so often in physical simulation this technique is addressed separately.

The finite element method is essentially the optimization of a system of linear or nonlinear constraints which are approximations to systems of partial differential equations or integral equations. Boundary value problems provide additional constraints on the boundaries which prevent degenerate solutions. The reasons for the finite element method's popularity stem from its flexibility and its extensive resource and support base. The method is well studied and many ready-to-use implementations are publicly available. However, there are some significant drawbacks to this popular technique. An FEM solution requires a well-placed mesh, and although there has been much work in this area (e.g., [55]), it continues to be a problem. In addition, because the systems of equations are generally nonlinear, the method is slow. Only in very simple cases in which the systems are extremely well behaved, sparse, or linear can solutions be obtained in interactive speeds[21, 38]. Finally, the finite element mesh elements are usually represented by facets because the differential equations are easier to form. Therefore the results of the simulation are often not acceptable within the framework of higher-order parametric surface design systems. There have been proposed integrations of higher-order parametric surfaces into the finite element world (e.g., [94]), but none of these solutions can support the fracture and other discontinuities which can easily result from the finite element method and are more easily represented by facets.

To summarize, the applications which would benefit by incorporating discontinuities within the representation are varied despite the fact that the design techniques for generating these situations are standard. However, the problems associated with representing discontinuities are not easily solvable by the prevailing methods. Tradeoffs are required to obtain adequate results depending on the application. In Chapter 3, capabilities of existing representations and previous attempts to address these issues are discussed.

CHAPTER 3

PREVIOUS WORK

Discontinuities introduced during the design process pose difficulties when using higher-order representations. Although the representation can be changed to accommodate the new structure, the design process is disrupted and the resulting representation may not support design operations similar to those available for the original representation. The approaches to solving this representational closure problem include the following:

1. Use as an original representation *a lower-order, generally faceted, representation* which is known to be closed under operations which introduce discontinuities.

2. Use as an original representation *a higher-order, usually parametric, representation* with enough degrees of freedom to represent simple cases. Degeneracies may be introduced to support various continuity features. Representation reverts to faceted representation once a certain complexity is reached.

3. Use as an original representation *a modification or extension of a higher-order representation* which is closed under a number of operations which introduce discontinuities.

The following section reviews the representation classes and how they attempt or could attempt to address the representation closure problem.

3.1 Faceted Representations

Faceted representations have the advantage that they are easy to understand and to implement. The primary disadvantage of the faceted representation is that the collection of facets only approximates smoothness to within some tolerance. If the tolerance is dramatically reduced, the size of the data structure may expand prohibitively. Manipulation of models represented by facets requires some knowledge of the characteristic smoothness of the surface at a higher level than individual facets. Ultimately, this is no different than the problem of representing discontinuities in a higher-order model. On the other hand, the topological flexibility for faceted representations is limited only by the size of the model. If a faceted representation is used for a physically based simulation, the representation will not need to dramatically change even if discontinuities are introduced[73]. Despite these advantages, higher-order surfaces are being used consistently in design systems and useful representations which are closed under these design processes are needed.

3.2 Parametric Surfaces

Parametric surfaces come in many flavors, the most popular of which are multivariate splines. The tensor product B-spline representation is a special case of multivariate splines and one of the more widely used parametric surface representations

3.2.1 Multivariate Tensor Product Splines

Each tensor product B-spline surface blend function is actually just the product of two univarite B-spline blend functions which govern the blending of control points to describe the surface. One of advantages of the tensor product B-spline representation is the fact that it is simple and that algorithms such as refinement and order manipulation are well known. Unfortunately, the support of tensor product surfaces is parametrically rectangular; therefore, any degenerate knot configuration which may contribute to a discontinuity is present along the entire parametric isoline. Most other multivariate splines fall into the category of alternate higher-order representations[22]. Unusual multivariate splines (such as Box splines[59]) do not necessarily have rectangular parametric domains, and so any degenerate knot sequence which creates a discontinuity although isoparametric in nature does not necessarily lie in a particular direction across the surface.

However, these other representations have yet to make a significant impact in design for several possible reasons. First, they are more difficult to understand so are less attractive for use in commercial products. Second, standard components present in other parametric surfaces such as simple basis functions and order relationships are not generally present for multivariate splines. This makes analysis for simulation and interactive design processes very difficult. In addition, the complexity and irregularity of the blending functions make this general class of representations slow. Finally, arbitrary discontinuity within the surface is generally not supported.

The tensor product torn B-spline surface representation presented in this thesis may also be applicable to general multivariate splines. This type of extension is left for future work.

3.2.2 Subdivision Surfaces

Subdivision surfaces are another class of surfaces which demonstrates potential for being able to represent surfaces of arbitrary topological type and embedded discontinuities. These surfaces are constructive surfaces based on a parameter and a control net corresponding to the connectivity of the base-level surfaces. The most common subdivision type are triangles[51, 46, 47, 45] although both quadrilaterals[28] and biquadratic and bicubic tensor product B-splines[19] have been used for the base-level surface type. The topological flexibility of these surfaces is striking, yet several things stand in their way to becoming the surface of choice in a design system. First, simulation and interactive manipulation of these surfaces are difficult because the surface is constructive. Specifically, it is not clear how discontinuities can be added to the model after the model is constructed simply because there may be no clear way to translate the information back to the control mesh. Current implementations deal only with surface reconstruction[46, 47, 45], not interactive manipulation. In addition, differential properties of these surfaces can be difficult to obtain since the mathematical formulation is constructive.

3.3 Alternate Parametric Surface Representations

Most of the development in alternate representations of parametric surfaces has used the general class of splines as the base. The development is spurred by implementations of Béziers, B-splines or NURBS in in popular commercial design systems (e.g., AutoCAD, SoftImage, EZFeatureMill, SurfCAM, 3DStudio). Changing the spline representation has been a popular technique, among the results are γ-splines or $G^{(2)}$-splines, β-splines[30], ν-splines or tension splines[61], G-splines[44], Box splines (a type of multivariate spline)[59], Hayes splines[41], and X-splines[11]. Only the last two have specifically addressed the representation of discontinuities within individual surfaces, but each at the expense of making it difficult to support the more common spline methods of refinement and subdivision. The most interesting, yet least useful, representation is Hayes splines[41]. Hayes splines provide a functional definition of the knot vector with respect to the opposing parametric value. Hayes splines can represent partial discontinuities in a parametric direction, although they need not be isoparametric. One of the obvious difficulties with this representation is the complexity imposed by this additional level of indirection. Hayes splines are hard to describe and substantially harder to use. Recently, X-splines[11] were introduced as a combination of B-splines and Catmull-Rom splines[30]. They appear to be easier to use than Hayes splines, although still providing partial discontinuity across a single spline surface, but the discontinuities are still isoparametric. In addition, the surface is not a true tensor product because of a normalizing factor, so common spline methods such as refinement and subdivision have different meanings.

The alternate representations are generally very complex and too difficult to control for widespread application. In addition, the number of operations for which the set of representable models are closed is quite small.

3.4 Trimmed B-spline Surfaces

Another modification often used with parametric tensor product surfaces like B-splines is a *trimmed* representation. A trimmed surface is a surface whose original domain has been restricted to a set of closed subregions of the original domain. Early techniques developed by Thomas[82] and Carlson[16, 17] approximated the trimmed surface within the restricted domain with a set of polygons. Sarraga[70] suggested using a collection of rational tensor product surfaces to approximate the trimmed surface. However, in each of these techniques the original surface representation is different from the trimmed surface representation, making modifications within the same framework difficult. Representing the trimmed region by an unevaluated two dimensional CSG tree was suggested by Casale[18] but evaluation of this surface can be slow and tedious. Representing the boundaries of the trimmed region by algebraic curves was suggested by Farouki[31], but this technique does not scale well to general tensor product B-spline surfaces. Another technique was developed at the University of Utah by McCollough[56]and uses a parametric curve evaluated in the domain of the surface to represent the boundary of the trimmed region (see Figure 3.1). Trimmed B-spline surfaces are often used in solid model boundary representations in which the boundary surfaces are nonrectangular. The computation of intersections of higher-order parametric surfaces such as B-splines is generally not tractable, and the resulting intersection curves are usually not representable by simple parametric curves embedded in the surfaces (see Figure 3.2). In McCollough's representation, the actual intersection is approximated by a piecewise linear intersection curve along with the corresponding parametric locations of the individual points within each of the surfaces (see Figure 3.3). One major disadvantage to this representation is that a large amount of data is needed to represent the boundaries of the trimmed region when an adjacency is involved.

At first it may seem that the trimmed B-spline representation would support general discontinuities, but there are serious drawbacks to using this representation. First, even though complex topologies can be represented, regions which appear independent by visual cues may not really be independent in the underlying

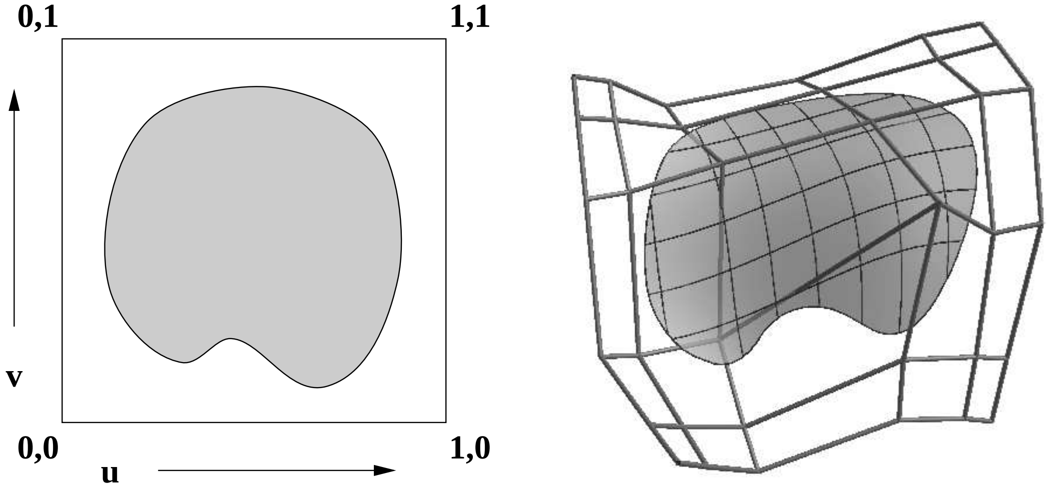

Figure 3.1. Example trimmed surface with parametric curve.

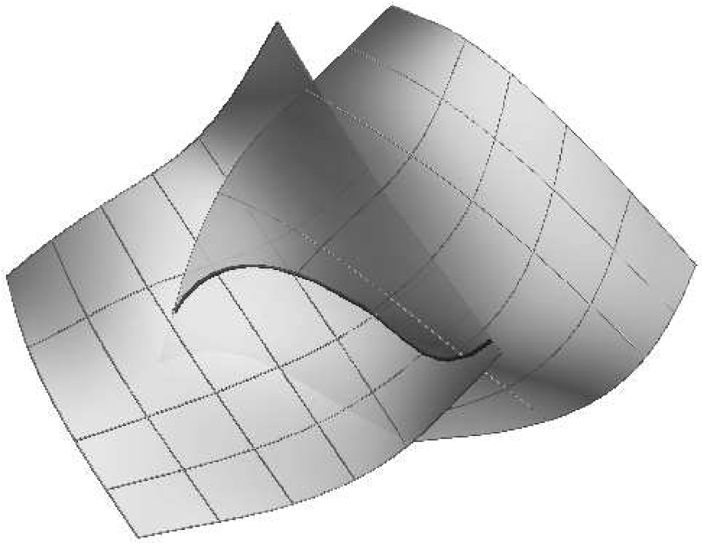

Figure 3.2. Piecewise linear intersection between two surfaces.

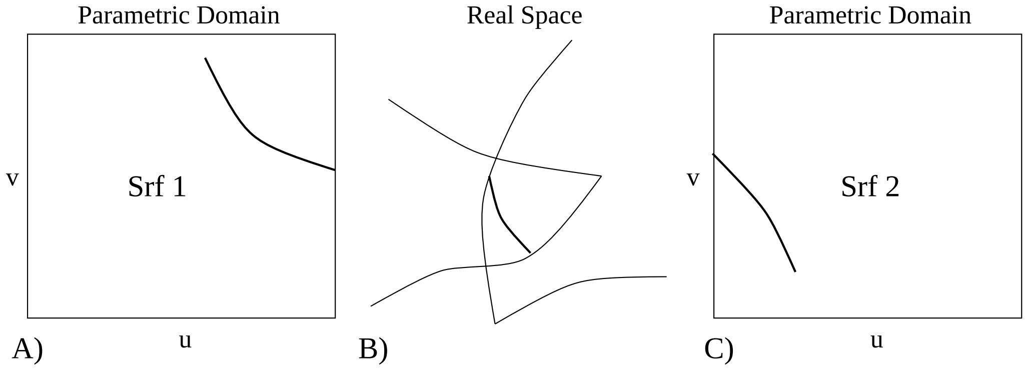

Figure 3.3. Three-way data structure for a trimming curve. A) Parametric curve in surface 1. B) Euclidean intersection curve. C) Parametric curve in surface 2.

representation.

Consider a U-shaped region cut from a uniform bicubic B-spline. Then the underlying surface,

$$S(u,v) = \sum_{i,j=0}^{4,6} P_{i,j} B_{i,\tau^{\mathbf{u}}}^{k}(u) B_{j,\tau^{\mathbf{v}}}^{k}(v) \tag{3.1}$$

where the order, k, is 4 (cubic in each parametric direction), the knot vector $\tau_u = \{0,0,0,0,1,1,1,1\}$ and the knot vector $\tau_v = \{0,0,0,0,1,2,3,3,3,3\}$. The knot vectors indicate that in the u direction, there is only one interval, $[0,1]$, but in the v direction, there are three intervals, $[0,1]$, $[1,2]$ and $[2,3]$. These intervals correspond to the piecewise polynomial patches (see Figure 3.4). A single patch of a bicubic B-spline has nonzero basis functions, (i.e., $B_{i,\tau^{\mathbf{u}}}^{k}(u)$ and $B_{j,\tau^{\mathbf{v}}}^{k}(v)$) for 16 control points. Unfortunately, a single patch represents the base of locality of a B-spline surface; every point within the patch is dependent on all 16 control points (patch boundaries excepted). Modifying any of the 16 control points corresponding to a particular patch will modify the shape of the entire patch.

Suppose the surface is trimmed so that only a U-shaped region remains (see Figure 3.5). If such a region were cut from a piece of paper, the two ends would be independently flexible. Intuitively, the same would be expected of the region cut from the B-spline surface. Notice that both ends of the U contain sections

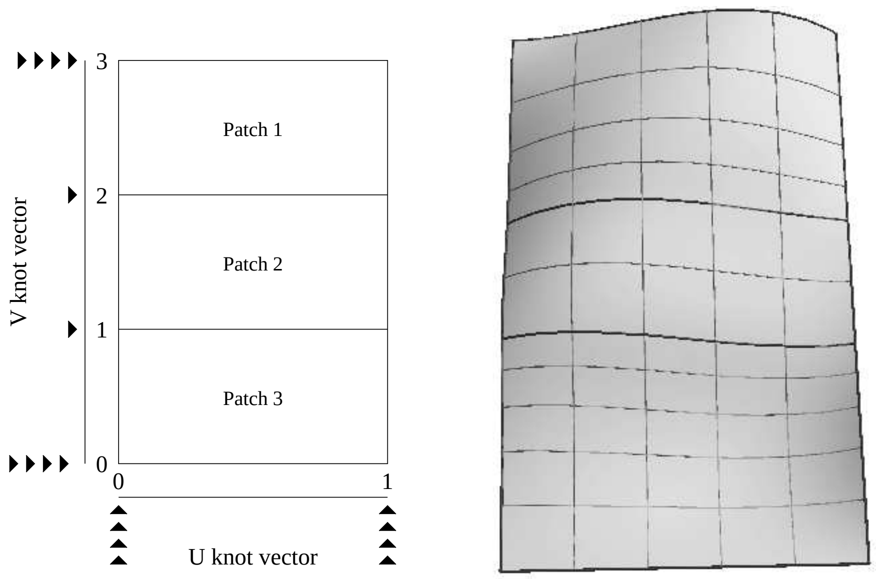

Figure 3.4. Initial surface with knot vectors and patches outlined.

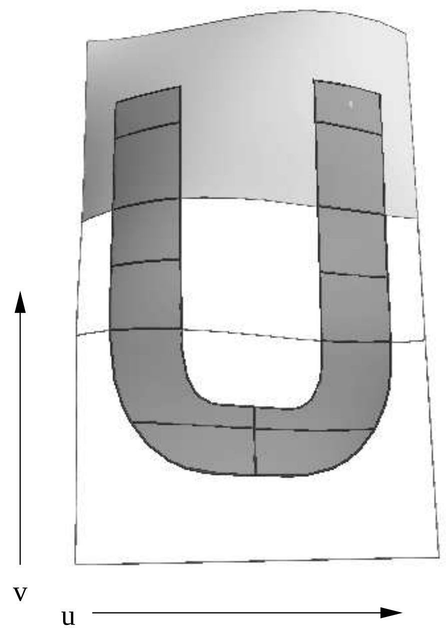

Figure 3.5. Ends of "U" are in the same patch.

from the same patch. Because of this, they are totally dependent on the same control points. If sufficient flexibility were added to the surface by introducing more knots and control points so that the individual patches in the two ends are totally independent of each other (i.e., the spans of the patches do not intersect), then this interdependence could be avoided. Unfortunately, reducing the distance between the two ends (see Figure 3.6(A)) results in a situation in which only subdividing the surface into two surfaces is sufficient to maintain the independence of the ends but subdivision reduces the continuity in the section of the U that crosses the subdivision boundary (Figure 3.6(B)) and is therefore not an acceptable solution.

This interdependence causes another difficulty. Regions are independent only if a sufficient amount of the surface is removed between the two regions. During a design process which introduces discontinuities, the parametric domain often represents a section of a physical model. Removing or trimming away a section of the domain represents removal of material which is often not acceptable.

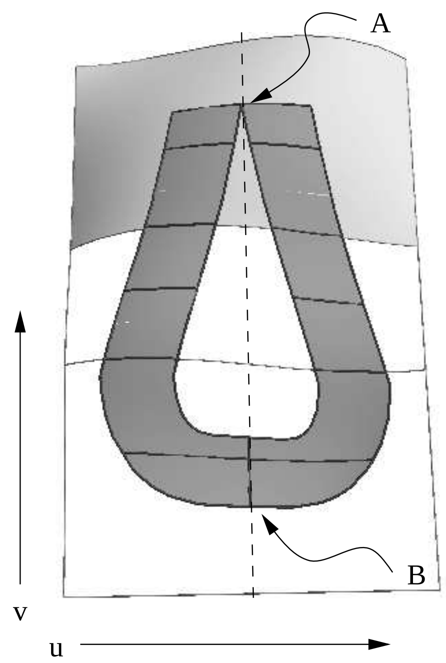

Figure 3.6. Dependence problems in trimmed surfaces.

3.5 Constraints

Regardless of the approach used to represent discontinuities, constraint fulfillment is often used to keep the model together[50, 8, 65, 92, 93, 25, 79]. In a typical design environment, the designer supplies adjacency or tangent requirements and then must convert these requirements into a set of constraint equations that can be numerically or analytically satisfied within the design environment.

The primary difficulties with this approach are the complexity of the constraint functions, the lack of a single coherent parametric space (if the discontinuities were introduced after the model was originally built)[33, 90], and the possibility of not being able to find a solution to the user-provided system of constraints due to the nonlinearity of the equations, over-constraining the system, or getting stuck in local minima. The advantages of this approach include its flexibility, its enormous popularity in current simulation methods (such as FEM[9]), and its large supporting base of research. The torn B-spline representation supports $C^{(-1)}$ continuity but creases require $C^{(0)}$ continuity. Unfortunately, exact solutions to this $C^{(0)}$ continuity requirement are often intractable or nonexistent. Therefore linear constraints are used to obtain an approximate solution in a reasonable amount of time. The particular approach used to construct the linear constraint equations is derived from the approach described by Fowler[35, 34].

3.6 Patching

Combining multisided smooth patches and triangular patches [19, 42, 53, 69, 52, 40, and others] into interesting models has been a long standing approach. The topological flexibility offered by this approach is enormous, but most of these techniques produce only uniformly smooth models. Recently, interest in models which have creases, tears, and other continuity features has increased and new techniques have been developed with support such features[74, 45, 83]. It can safely be said that each of these new techniques also uses a multipatch scheme, in which a significantly large set of basic building blocks are combined with constraints or some other smoothing method to produce a smaller number or smoother set of patches to produce the desired continuity features in the larger model. A recent example [45] begins with a set of triangles and through optimization and feature recognition produces a piecewise smooth representation with the continuity features intact. An earlier example, in computer vision reconstructs surfaces with discontinuities[74]. In contrast to the multipatch approach, the torn B-spline introduces continuity features within a smooth surface.

3.6.1 Topology

The torn tensor product B-spline surface dramatically expands the representational capabilities of a single tensor product B-spline surface. Several common topological terms will be used in this thesis and are defined below[60].

Definition 3.1 *If U is an open set containing x, then U is said to be a* neighborhood *of x.*

Definition 3.2 *A* separation *of a space X is a pair U,V of disjoint nonempty open subsets of X whose union is X.*

In particular, its opposite, *connectivity*, is critical in determining the necessary continuity requirements in the surfaces surrounding tears.

Definition 3.3 *An* m-manifold *is a space X, such that each point x of X has a neighborhood that can be mapped 1-to-1 and onto an open subset of $\mathbb{R}^m$.*

Most boundary representations of solid models are 2-manifold. Tears in a smooth boundary representation make the boundary representation nonmanifold.

Although most data representations theoretically permit nonmanifold topology when combining more than one basic building block, it is rarely used since most real objects have manifold boundary representations. With the increased use of visualization and simulation in the physical sciences and mathematics, the use of nonmanifold topologies has also increased. Bloomenthal and Ferguson[12] use triangles to represent their topology in a recent treatment of nonmanifold topology for implicit surfaces.

The torn B-spline surface described in this thesis allows discontinuities to be introduced into the surface, causing a model to become nonmanifold. Treatment of this situation is given in Section 7.3.

The representations and modeling techniques described in this chapter are the state-of-the-art for representing discontinuities in models. Yet despite their capabilities, these representations and techniques are difficult to describe and use and often fail to meet the flexibility requirements of real-world situations which are becoming increasingly more common. Clearly a flexible representation is capable of representing arbitrarily complex discontinuities without information loss in other areas of the model and is easy to understand without adding undo complexity to the representation. In the following chapter, the torn tensor product B-spline surface representation is presented. The torn B-spline surface provides the flexibility without information loss that is necessary for embedding discontinuities in the inherently smooth surface representation.

CHAPTER 4

TORN B-SPLINES

The *torn B-spline* representation is built upon the well-known B-spline representation. It uses some key techniques from the trimmed B-spline representation to provide some of the most useful functionality of the B-spline class, such as evaluation and display. In addition, modeling techniques such as designing with *feature curves* and *constraints* are integral parts of this representation. The previous research on incorporating discontinuities within models was reviewed in Section 3.3. Section 4.2 contains the technical foundation for torn B-splines, including definitions for the B-spline representation and other core definitions. Section 4.3 introduces the torn tensor product B-spline. In the following section, several types of discontinuities are presented to lay the foundation for the rest of the chapter. It is these types of discontinuities that the torn B-spline surface is able to represent.

4.1 Continuity Features

The examples in the previous sections present several situations in which discontinuities arise within the context of the modeling process. These discontinuities characterize the model or differentiate the model from other models. Therefore they are called *continuity features*.

What is a *continuity feature*? By word analysis, a *feature*, according to Webster, is "a prominent or conspicuous part or characteristic"[26, p. 487, definition 1]. *Continuity* in this sense, refers to the smoothness of the model. The definition a continuity feature is as follows.

Definition 4.1 *A* continuity feature *is a characteristic change in the continuity of a model.*

Continuity features are often the main focus of an outline drawing of a model since they characterize the shape of the model. The vertices and edges of polygons, curved boundaries of a sculpted surface, and folds in material are all continuity features. In a design system, the continuity features are represented by vertices, edges, or curves, all defined within the context of the higher-order modeling constructs such as surfaces or solids. In most situations, these features are "well behaved" in that they form boundaries of the higher-order elements used to construct the model. Occasionally, these features are not a natural part of the boundary (i.e., they are not part of a closed loop which would form a boundary in a surface model) and so are not easy to compute or use. The designer must then switch representations or perform some additional work in order to adequately represent this type of feature.

The continuity features introduced or discussed in this thesis are defined below.

Definition 4.2 *A* tear *is a smooth parametric curve in the parametric domain of a surface along which the surface is geometrically discontinuous (see Figure 4.1(A)).*

Definition 4.3 *A* crease*[45] is a smooth parametric curve in the parametric domain of a surface along which that surface has $G^{(0)}$ continuity but not $G^{(1)}$ continuity (see Figure 4.1(B)).*

Although these features are defined in terms of geometric continuity, it is assumed that the parametric surface is standard and has no other constraints unless noted so that these features can be discussed in terms of parametric continuity.

Definition 4.4 *A* critical point *is the endpoint of a tear in the interior of a surface, whether or not it is part of a crease. An endpoint on a surface boundary is not critical unless the boundary is constrained to be adjacent to a boundary of another surface in the neighborhood of that point (see Figure 4.1(C)).*

Definition 4.5 *A* dart*[45] is a crease whose endpoint lies in the interior of a continuous surface (see Figure 4.1(D)).*

If the interior endpoint of the dart is also the endpoint of a tear, then that endpoint is a *critical point*. In particular, the continuity characteristics change at the end of a dart.

The same smoothness characteristics are required for all points on the surface except for the points on the tear and the critical points. A critical point is the only point on a tear whose connected neighborhood contains both sides of the tear. These critical points play an important role in the torn B-spline data structure.

Definition 4.6 *A* corner*[45] is the point at which two or more creases join. Usually a corner describes a $C^{(1)}$ discontinuity between the parametric representations of the creases within the surface (see Figure 4.1(E)).*

This term can also be used for the intersection of two or more tears, although the resulting geometry is quite different. A more general definition of a corner could allow a corner to exist in the middle of a single crease, but given the assumption that a single curve is $C^{(1)}$ with respect to parametric space of the surface, corners require two or more curves.

Definition 4.7 *A* fracture *is a pair of parametric surfaces embedded in a solid which represents a discontinuity within the solid (see Figure 4.1(F)).*

Practically speaking, any parametric surface slice which intersects the fracture will have a tear in the surface along the intersection. In the engineering world, a fracture is one of the most complex continuity features used. This thesis introduces a method for representing the parametric slices of the solid, and issues involved in representing the solid itself will be discussed.

4.2 Technical Background

First the basic definition of a tensor product B-spline surface and some additional terms which are frequently used will be given.

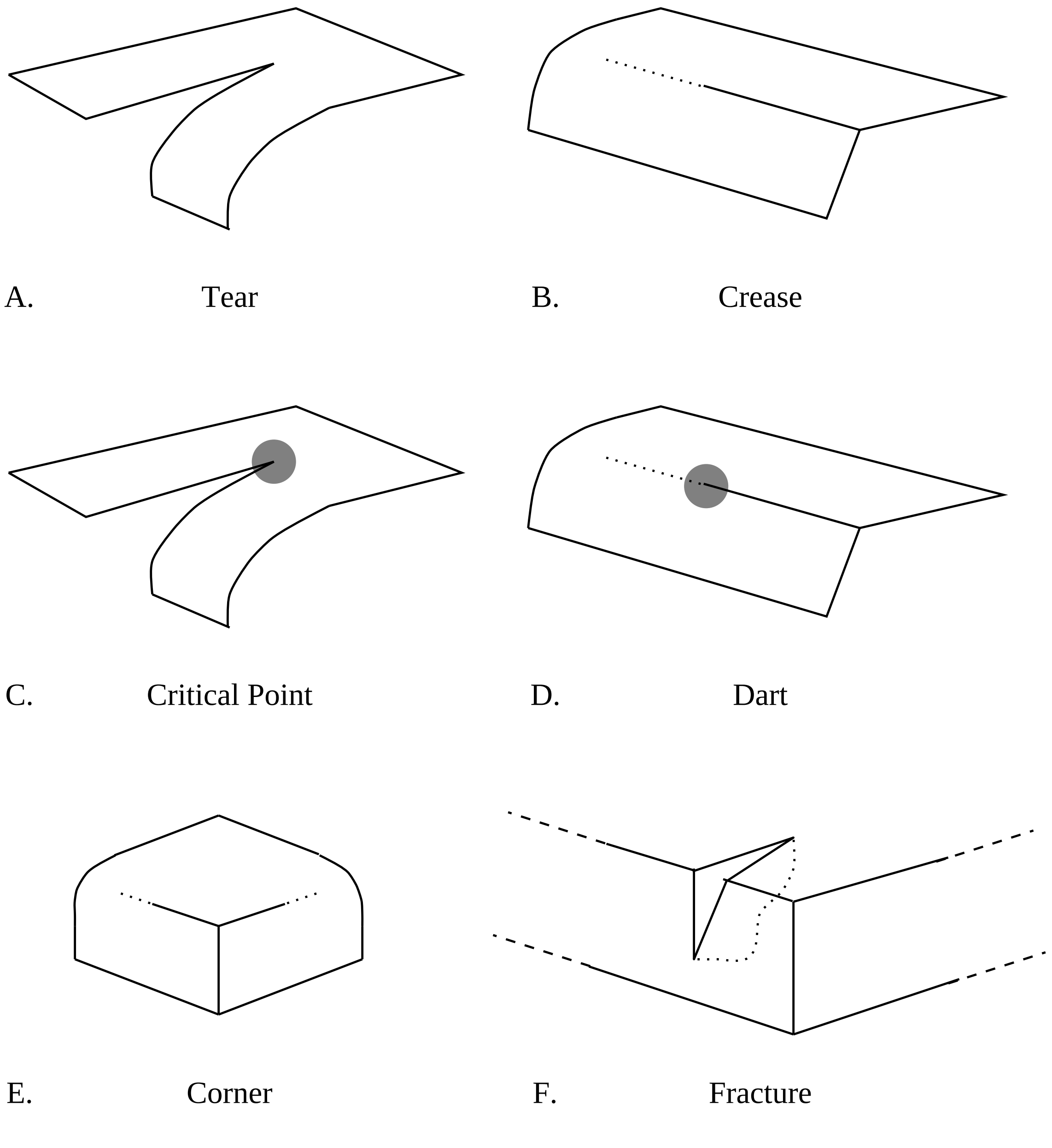

Figure 4.1. Continuity features.

4.2.1 Tensor Product B-spline

Definition 4.8 *A nonuniform tensor product B-spline surface,*

$$Q(u,v) = \sum_{i,j=0}^{m,n} P_{ij} B_{i,\tau^{\mathbf{u}},k_u}(u) B_{j,\tau^{\mathbf{v}},k_v}(v), \tag{4.1}$$

is defined by the set of coefficients, $\{P_{ij}\}$*, the knot vectors,* $\tau^{\mathbf{u}} = \{\tau_i^u\}, \tau^{\mathbf{v}} = \{\tau_i^v\}$*, and the B-spline basis functions,* $\{B_{i,\tau^{\mathbf{u}},k_u}(u),\ B_{j,\tau^{\mathbf{v}},k_v}(v)$ *where* $\{B_{i,\tau^{\mathbf{u}},k_u}(u)$ $(B_{j,\tau^{\mathbf{v}},k_v}(v))$ *is the* $i^{th}(j^{th})$ *B-spline basis function of order* $k_u(k_v)$ *over the knot vector* $\tau^{\mathbf{u}}(\tau^{\mathbf{v}})$*, respectively.*

To simplify notation where there is no confusion, the knot vector and the order will be inferred from the parametric variable that is used, so $B_{i,\tau^{\mathbf{u}},k_u}(u) = B_i(u)$ and $B_{j,\tau^{\mathbf{v}},k_v}(v) = B_j(v)$. In addition, the product of the basis functions will be abbreviated as $B_{ij}(u,v) = B_i(u)B_j(v)$. To distinguish the torn B-spline definition from the above definition, a surface from Definition 4.8 will be referred to as the *standard* tensor product B-spline surface.

4.2.2 Span

Definition 4.9 *The* span, $\mathcal{S}(u,v)$*, is the set of (ordered pairs of) subscripts whose corresponding basis functions are nonzero at* (u,v)*, i.e.,*

$$\mathcal{S}(u,v) = \{(i,j) | B_{ij}(u,v) \neq 0\}. \tag{4.2}$$

Definition 4.10 *Let* $q(t) = (u(t), v(t))$ *be a parametric curve,* $t \in [t_{min}, t_{max}]$*, in the parameter space of a surface* Q*. The* span *of a curve,* $\mathcal{R}(q)$*, is defined as the union of the set of (ordered pairs of) subscripts whose corresponding basis functions are nonzero for some* $(u(t), v(t))$ *on the curve,* q*, i.e.,*

$$\mathcal{R}(q) = \bigcup_{t \in [t_{min}, t_{max}]} \mathcal{S}(q(t)). \tag{4.3}$$

The span will be used to determine which control points are crucial for determining the set of parametric values along a continuity feature.

4.2.3 Patch

Definition 4.11 *The* patch *of a span, $\mathcal{G}(\mathcal{S})$, is the closure of the set of (u, v) all of which have the same span.*

In general, if $\mathcal{S}_1$ and $\mathcal{S}_2$ are two spans of either points or curves, and $\mathcal{S}_1 \subset \mathcal{S}_2$, then $\mathcal{G}(\mathcal{S}_1) \subset \mathcal{G}(\mathcal{S}_2)$.

The knot lines (or interior knot values) of the B-spline definition delineate the patches of the surface. For the torn B-spline representation, the continuity characteristics may not change within a patch.

4.3 Torn B-splines

Consider the process of introducing discontinuities into a model given the tools that are currently available. There are two natural ways to think of this process. First, basic modeling elements can be put together in such a manner that the discontinuity lies along the boundary between elements. Where the model needs to be smooth, constraints can be used to enforce some degree of smoothness. This is a "bottom-up" approach. The second way is to arrange the parameters of the basic modeling element so that a discontinuity is formed within the modeling element itself. For example, a tensor product B-spline can have multiple knots, each knot lowering the degree of the surface by 1. This is the "top-down" approach.

The "bottom-up" approach is easy to understand. However, since the surfaces may be parameterized differently, this may become a highly nonlinear constraint problem, one that may not be solvable. In addition, individual surfaces behave independently, requiring highly specialized code to make modifications to the region as a whole. Finally, if discontinuities were added to a smooth surface, the original surface's structure provides a wealth of smoothness information away from the discontinuity that is thrown away when multiple surfaces are constructed. The "top-down" approach is more desirable for these reasons. Unfortunately, current representations are limited in their flexibility for representing discontinuities; therefore the "top-down" approach is limited to certain representations in certain situations, the merits of each were discussed earlier.

The bottom-up approach to accommodating discontinuities within a single tensor product B-spline representation results in a set of trimmed B-spline surfaces, each with its own set of coefficients (control mesh) and a set of constraints which *tie* the trimmed regions back together along the smooth parts, but the constraints along the smooth parts are just composed of matching coefficients since both trimmed regions came from the same surface. The torn B-spline representation is based on this concept as applied to the "top-down" approach. An exact solution for smoothness away from the discontinuities is guaranteed implicitly and the parameterization matches across the discontinuities by default by focusing on supporting the discontinuity within the structure instead of continuity between two different structures. In addition, the surface is one complete unit with full knowledge of all its domain and is capable of supporting operations on the surface as a whole.

The development of the torn B-spline representation from the "top-down" approach is best understood by first considering the two-dimensional (or 2D) case of a torn B-spline curve. Figure 4.2 shows a torn B-spline curve. To tear a B-spline curve, q, we introduce a *tear point* at parametric location, $\hat{t}$. Note that $\mathcal{R}([t_{min}, \hat{t}]) \bigcap \mathcal{R}([\hat{t}, t_{max}]) = \mathcal{S}(\hat{t})$. Let η be a classification function which separates the curve into two distinct regions, $[t_{min}, \hat{t}]$ and $[\hat{t}, t_{max}]$. (Although $\hat{t}$ does not actually exist in both regions, we assume the limiting case.) $\mathcal{S}(\hat{t})$ then occurs in the spans of both regions. If q is torn at $\hat{t}$, then there must be two distinct locations for $q(\hat{t})$. In order to make these two locations independent of each other, their spans must be independent; hence, the torn representation for each region must have distinct control points for each i in $\mathcal{S}(\hat{t})$. These additional points are called the *overlap polygon.* In Figure 4.2, row a contains the original control polygon of a curve. Row b contains the set of control points, Q3-Q6 and P7, for $\mathcal{R}([\hat{t}, t_{max}])$, and row c contains the set of control points, P0-P6, for $\mathcal{R}([t_{min}, \hat{t}])$. The function, η, then determines which curve region a parametric location is contained in. A point on the curve, $q(t)$, is evaluated by using the appropriate control points from the original control polygon and the overlap polygon as appropriate for the region of the curve which contains $q(t)$ as determined by η.

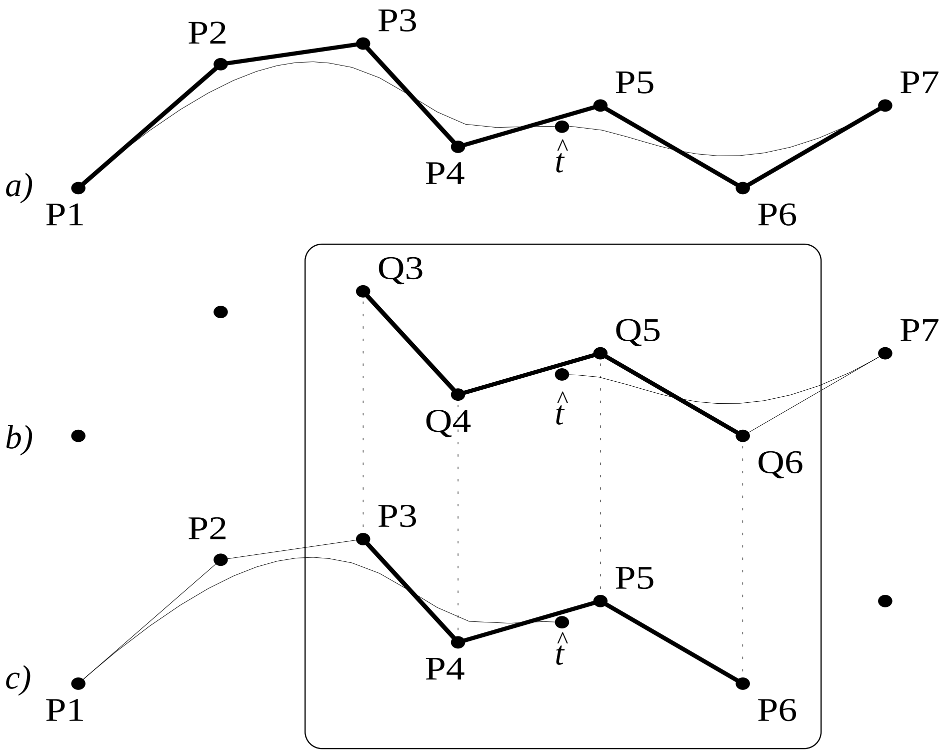

Figure 4.2. Torn B-spline curve.

In the curve case, the two curve regions can be made independent by subdividing the curve at the tear point. Since subdivision is a well-established technique, the torn B-spline curve is mostly academic. The torn B-spline surface is much more interesting. Informally, a torn B-spline surface is comprised of an underlying tensor product B-spline surface and one or more curves of discontinuity or *tear curves*. Consider first, the case where a single tear curve, $q(t)$, separates a surface into two distinct regions. Again the requirement is that the surface regions on either side of the tear curve be independent of each other. The two regions are again classified by the function, η. In order for the regions to be independent of each other, the span of the tear in one region must be independent of the span of the tear in the other region, which is analogous to the torn B-spline curve case. That is, a point on the tear in one region must use a completely different set of coefficients for the span of that point. This implies that $\mathcal{R}(q)$ in one region is independent of $\mathcal{R}(q)$ in the other region. The additional coefficients required to make the spans independent are stored in the *overlap mesh*. Figure 4.3(A) shows an isoline drawing of the torn B-spline surface. Figure 4.3(B) shows the patches which contain the tear curve. Figure 4.3(C) shows the points in the span of the tear curve, $\mathcal{R}(q)$, which are included in the overlap mesh. Once again, each region, separated by η, is associated with a particular set of control points selected from the original mesh and the overlap mesh. Although the maximal number of degrees of freedom (DOFs) associated with the discontinuity is fixed by the configuration of tear curves in the surface and the separation of regions (see Section 4.7), the methods used to determine the distribution of these points are implementation dependent (see Section 4.10.7).

Unfortunately, tear curves which fully separate the domain are not the most common case, since full separation is often representable by other methods (i.e., subdivision or trimmed surfaces). More typically, one or more of the endpoints of the tear curve is in the middle of the surface; often in the middle of a patch as well. These partial tears are related to Thomas' *cut curves*[82]. When the tears separate the surface, it is clear how to determine the classification function, η. When the

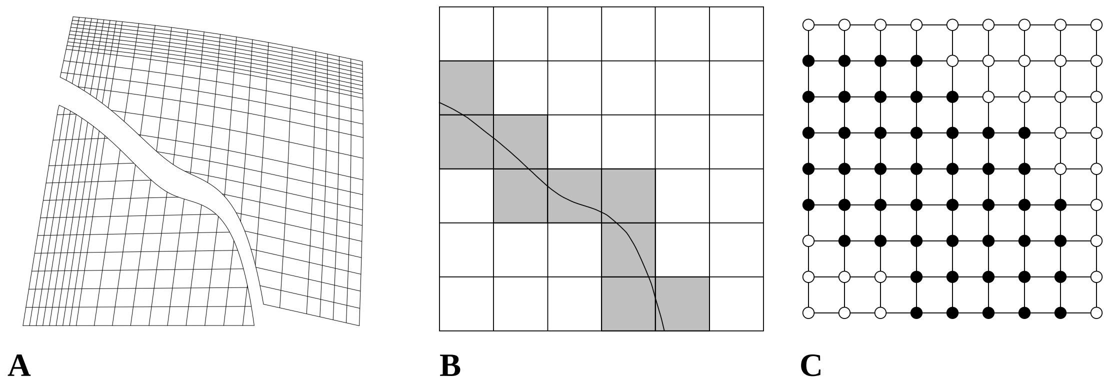

Figure 4.3. Torn B-spline surface with complete tear.

tears do not separate the surface, the distinction is no longer clear. Fortunately, the original smoothness of the surface is required away from the discontinuities; thus only one surface location is associated with a given parametric location. Regardless of how the regions are classified, the evaluation of a given point must be well defined in the context of the torn surface. A method for making this classification well defined is presented in Sections 4.7 and following.

If a tear curve has an endpoint in the middle of the surface, the neighborhood of this critical point has an assumed continuity requirement; the surface must have the same continuity as the underlying B-spline surface except possibly at the tear. If the tear has an endpoint in the middle of a patch, the entire patch must remain connected. Recall that all points in the interior of the patch have the same span; therefore they are all dependent on the same coefficients. This dependence because the cross-patch continuity is being maintained on all patch boundaries and the span is essentially the smallest piece of the surface which is able to support the cross-patch continuity described by the order and knot vector of the underlying tensor product B-spline surface. This characteristic of the patch means that if two points are connected in the same patch, then there is only one configuration of coefficients given the boundary continuity requirements for that patch. Conversely, if the spans of two parametric locations are different in at least one coefficient, then the two points are not connected within the patch. Therefore, either the patch is separated from one patch boundary to another, or any two points on the patch are connected and the patch retains its original continuity. If the endpoint of the tear lies on a patch boundary, then this connectivity of the patch is not a problem, since the discontinuity crosses the entire patch. However, if the endpoint lies in the middle of a patch, there are several options available, the choice of which ultimately determines the degree to which the patch is kept smooth without modifying the tear itself.

1. Require that the entire patch containing the endpoint remains connected (continuous).
2. Allow the patch to become separated beyond the original tear curve description.
3. Introduce additional flexibility which creates a patch boundary at that point.

If maintaining smoothness where there is no discontinuity is more important than having full discontinuity along the entire length of the tear curve, then the first option is preferable to the second. This choice also prevents artifacts of the classification scheme from being present in the surface since the additional portion of the patch that is discontinuous from the second option is dependent on the classification of the regions. The third option was investigated, and initial experimentation indicated that it was a viable option also. Full integration and analysis of the effects of such a change are left for future work. An example of partial tear is given in Figure 4.4. Figure 4.4(A) is an isoline drawing of the surface. Figure 4.4(B) shows the patches of the surface that the tear extends through the hatched patch is the patch whose span is removed to satisfy the first option given above. Figure 4.4(C) shows the resulting control points that are in the overlap mesh.

Since the initial overlap mesh computation is derived directly from the span of the tear curve, the span of the tear's mid-surface endpoint is included in the overlap mesh. However, regardless if the tear ends on the boundary of a patch or in the interior of a patch, if any coefficient in the span of the endpoint is in the overlap mesh,

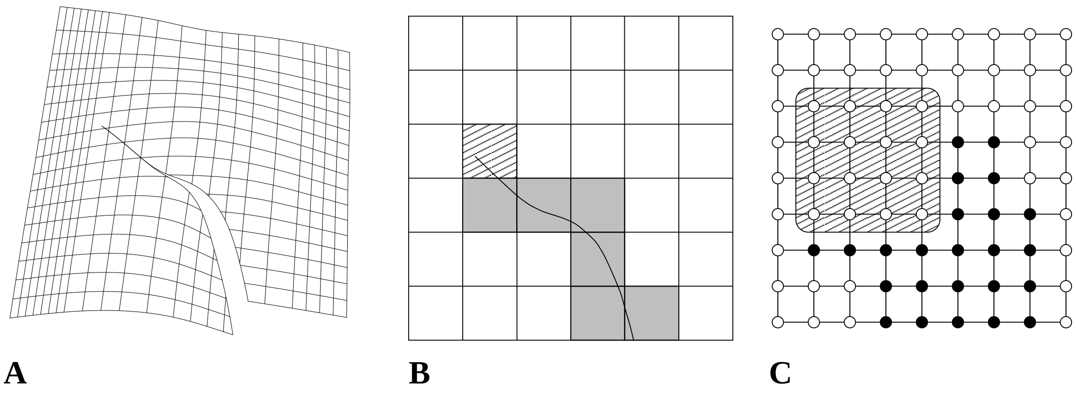

Figure 4.4. Torn B-spline surface with partial tear. A. Isolines from the surface. B. Parametric domain with the tear's span highlighted. C. Control points of the surface, highlighting the control points used in the overlap mesh of the tear.

the discontinuity extends across the patch. So the span of the endpoint is removed from the overlap mesh to maintain the continuity of the surface at the endpoint. In Figure 4.5(a), shaded regions of patches correspond to spans that have been added; the outlined regions (circled) correspond to spans which are subsequently removed. In Figure 4.5(b), the shaded points of the corresponding control mesh are added, and, likewise, the outlined points are removed. The remaining points in the darkly shaded regions make up the overlap mesh. Additional considerations for adding and removing points from the overlap mesh are discussed in Section 4.7.

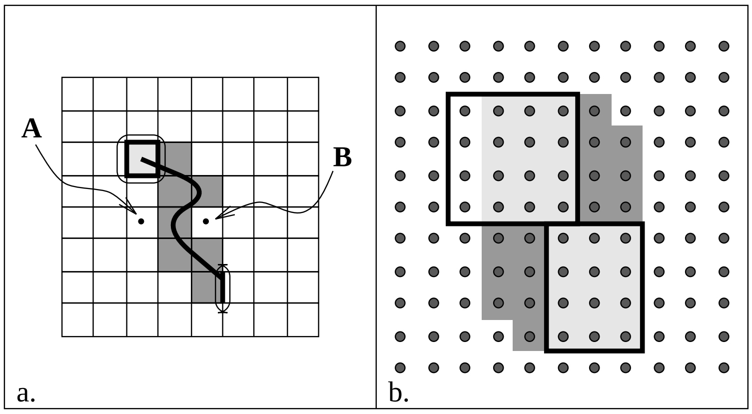

Figure 4.5. Subpatch diagram with tear.

4.4 Torn B-spline Definition

Definition 4.12 *The* torn B-spline surface *representation has several parts:*

1. *an underlying tensor product B-spline surface definition, with the requisite order, knot vectors, and control mesh;*
2. *a set of tear curves, $\{\gamma_\kappa\}_{\kappa=1}^{T}$, defined within the parameter space of the underlying B-spline surface, along which the surface is discontinuous;*
3. *a set of control meshes, $\{O_{ij}^{(\kappa)}\}_{\kappa=1}^{T}$, called* overlap meshes, *which contain the additional coefficients,*
4. *a masking function, μ, which identifies for each region, c, the composition of control points from the original control mesh, P_{ij}, and the overlap meshes, $O_{ij}^{(\kappa)}$,*
5. *a piecewise mapping, η_c, whose value is 1 if a given parametric point, (u, v), is contained in the parametric region, c, and 0 otherwise. This function may be ambiguous if the point lies on a tear curve or its extension.*

Then we define the torn B-spline *surface as*

$$T(u,v) = \sum_{c=0}^{T} \eta_c(u,v) \sum_{i,j=0}^{m,n} P_{ij}^{(c)} B_{i,j}(u,v) \tag{4.4}$$

where

$$P_{ij}^{(c)} = \begin{cases} P_{ij} & \text{if } \mu_c(i,j) = 0 \\ O_{ij}^{(\kappa)} & \text{if } \mu_c(i,j) = \kappa \text{ otherwise.} \end{cases} \tag{4.5}$$

For utility, let $\hat{\eta}(u,v) = c \iff \eta_c(u,v) = 1$.

Let us examine each of these parts in turn.

4.5 Underlying B-spline Surface

The underlying tensor product B-spline surface is the basis for the torn B-spline surface. The order and knot vector of the torn B-spline surface are inherited from it, and its control mesh serves as the base control mesh of the torn B-spline surface.

4.6 Tear Curves, γ_κ

As discussed in earlier sections, the tear curves which are part of the torn B-spline representation can have arbitrary parametric geometry and extend completely or partially across the surface. They may abut, but not cross, each other or themselves. The type of curve is limited only by the implementation. There is no limit on the number of tear curves in a torn B-spline surface.

4.7 Overlap Mesh, O

The additional DOFs allowed by the introduction of tears can be computed by examining the span of the tear.

Definition 4.13 *The* overlap mesh*, $O_{ij}^{(\kappa)}$, is the mesh of additional coefficients associated with the tear, γ_κ. By definition, $O^{(0)} \equiv P$.*

There are several requirements which make the classification of points a well-defined process. The first requirement is that the surface must retain its original degree of smoothness except along tear curves. Before the discontinuity is introduced, the knot vector describes the parametric continuity of the surface. In particular, for a tensor product B-spline surface of order 4 in the u direction, the surface will be $C^{(3)}$ in regions between knots and $C^{(3-m)}$ at knots of multiplicity m with respect to the u parameter. After the introduction of the tear discontinuity, the same degree of parametric continuity is maintained everywhere on the surface, except at the discontinuity. The second requirement is that the surface is discontinuous along the tear curve except where, to be so, would violate the first requirement. Finally, all additional degrees of freedom contained in the overlap mesh of each tear must be present in the new representation and must be independent. In a quick jump ahead to Section 4.10.6, when more than one tear is in a surface, the classification of the regions and the automatic distribution of the new control points may make two independent regions dependent, dropping conflicting degrees of freedom from the picture. This case eliminates certain choices for region classification and control point distribution.

4.7.1 Maximal Independence

When a tear curve is added to a B-spline surface, the tear curve makes the two regions on either side of the tear independent of each other. Depending on the separation of the domain into parametric regions, the number of additional degrees of freedom may be different. It can be proven that the additional coefficients present in the overlap mesh are both necessary and sufficient to make two points on opposite sides of the tear independent. However, in some cases, this independence is not the only independence required. In particular, if two points on the same side of the tear are to be independent of each other, the proof given below does not apply. In this case, the implementation of the torn B-spline data structure determines whether or not this independence is made available. In any case, for any given implementation and configuration of tear curves, there is a *maximal number of degrees of freedom* which can be utilized. This number is obtained by summing up the degrees of freedom in the original surface plus all the degrees of freedom in the overlap meshes. This concept, although it is implementation dependent, will be used throughout the remainder of this thesis.

However, it is possible to determine the globally maximal number of degrees of freedom for a given tear configuration regardless of implementation. A very simple heuristic can be used to determine the maximal number of degrees of freedom for a given tear. The algorithm essentially counts the number of times per coefficient that the curve doubles back on itself. In short, the curve is divided into *sections* where all of the points in the same section have the same span. We proceed along the curve, piece by piece, using the span of the piece as a "window" of sorts, marking the coefficients with a *tag*. After all coefficients have been marked for a given section of the curve, the tags of the coefficients in the window are all nonzero and negative; the tags for the coefficients outside of the window are zero if they have not been seen and greater than zero if they have been seen. The ordinal value of the tag indicates the number of times the window has "revisited" the coefficient. The detailed algorithm is presented below.

1. Identify sections of the tear curve according to spans, where all the points in a

section have the same span. (Only connected points can be in a given section, and points on patch boundaries make their own sections).

2. Initialize an array of integers to 0, one for each coefficient in the control mesh. Call this array the *tag array* and each item the *tag* of the coefficient.

3. Begin at one end of the tear and proceed through all of the sections consecutively, doing one of the following for each coefficient in the span of the section.

   (a) If the coefficient's tag is 0, set it to -1.

   (b) If the coefficient's tag is less than 0, do nothing.

   (c) If the coefficient's tag is greater than 0, increment it and negate it (i.e. 2 becomes -3).

   For each coefficient not in the span of the section, if the coefficient's tag is less than 0, negate it. Proceed to the next section.

4. After all sections have been processed, for each coefficient, if the coefficient's tag is greater than 0, negate it.

5. For each coefficient in the span of each endpoint (unless the endpoint lies on a boundary such as another tear, a trimming curve, or the boundary of the parametric domain), if the coefficient's tag is greater than 0, decrement it.

The resulting tag for each coefficient indicates the number of additional unique degrees of freedom required for maximal independence of the tear curve at that coefficient. Although the algorithm is simple, it may not be clear how best to use this information, since, at this point, it is a heuristic. However, it may affect how the tear curve should be split and may ultimately determine the best implementation. These and other related questions are left for future work.

4.7.2 Sufficiency of $O_{ij}^{(\kappa)}$

The span of the curve is used to compute the additional coefficients (DOFs) needed in the overlap mesh. Do these additional DOFs provide enough flexibility to cause a given tear curve to be discontinuous along its length? This question can be answered by carefully examining the construction of each O_{ij}. The necessity of each of the additional DOFs in the overlap meshes is demonstrated by the fact that in order for the parametric regions on either side of a parametric curve embedded in a given B-spline surface to be independent, the spans of the curve in each region must be independent of each other. To impose the requirement that the neighborhood around the ends of the tear curves have the same continuity characteristics as the original surface, the spans of the endpoints must be totally dependent on the same coefficients.

Are more coefficients necessary to describe the surface on either side of the tear? To show that these are the only coefficients needed, we first show that if a point lies between two other points then the intersection of the spans of the two other points is a subset of the span of the first point.

Theorem 4.1 *Let points A, B, and C on a B-spline surface of order k_u and k_v in the u and v directions, respectively, be described by parametric locations, (u_A, v_A), (u_B, v_B), and (u_C, v_C), respectively. Let C be* between *A and B if $min(u_A, u_B) \leq u_C \leq max(u_A, u_B)$ and $min(v_A, v_B) \leq v_C \leq max(v_A, v_B)$. Recall that $\mathcal{S}(p)$, is the span of the point, p. Then if $P_{\bar{i},\bar{j}} \in \mathcal{S}(A) \cap \mathcal{S}(B)$ then $P_{\bar{i},\bar{j}} \in \mathcal{S}(C)$.*

Proof. Assume, for now, that A, B, and C do not lie on knot lines. Let

$$\begin{aligned}
\mathcal{S}(A) &= P_{[i_A,\ldots,i_A+k_u-1],[j_A,\ldots,j_A+k_v-1]} \\
\mathcal{S}(B) &= P_{[i_B,\ldots,i_B+k_u-1],[j_B,\ldots,j_B+k_v-1]} \\
\mathcal{S}(C) &= P_{[i_C,\ldots,i_C+k_u-1],[j_C,\ldots,j_C+k_v-1]}.
\end{aligned}$$

Consider first the argument for the row index i; the argument for j follows analogously. If $P_{\bar{i},\bar{j}} \in \mathcal{S}(A) \cap \mathcal{S}(B)$, then

$$max(i_A, i_B) \leq \bar{i} \leq min(i_A, i_B) + k_u - 1.$$

Since C is between A and B,

$$min(i_A, i_B) \leq i_C \leq max(i_A, i_B) \tag{4.6}$$

$$min(i_A, i_B) + k_u - 1 \leq i_C + k_u - 1 \leq max(i_A, i_B) + k_u - 1. \tag{4.7}$$

Therefore,

$$i_C \leq \bar{i} \leq i_C + k_u - 1$$

which with the analogous argument for $\bar{j}$, implies that

$$P_{\bar{i},\bar{j}} \in \mathcal{S}(C) = P_{[i_C,\ldots,i_C+k_u-1],[j_C,\ldots,j_C+k_v-1]}.$$

As stated above, it is assumed that the points do not lie on knot lines. This implies that the number of indices for a given direction that are in the span of a point is equal to the order in that direction. If A or B lies on knot lines (at the minimum in either direction), the spans become smaller, but because the restriction becomes tighter, the argument still holds. However, if C lies on a knot line in a given direction, the span of C is reduced by the number of knots at that parametric location. This implies that Equation 4.7 does not follow directly from Equation 4.6 without additional argument.

Suppose then that u_C lies on a knot with multiplicity m. Then the number of indices in the span of C in the u direction is $k_u - 1 - m$ and the maximum index for i in the span of C is $i_C + k_u - 1 - m$. We need to show that $min(i_A, i_B) + k_u - 1 \leq i_C + k_u - 1 - m$ (the right-hand side of Equation 4.7 holds if $m \geq 0$). By the definition of the B-spline basis functions, the number of knots between the parametric locations of two points determines the difference between the beginning index values of the spans of the two points. This means that

$$min(i_A, i_B) + m \leq i_C.$$

Then,

$$min(i_A, i_B) \leq i_C - m,$$

and so

$$min(i_A, i_B) + k_u - 1 \leq i_C + k_u - 1 - m,$$

which is what we needed to show.

The argument is similar if $max(i_A, i_B)$ or $max(j_A, j_B)$ lies on a knot line. ∎

Definition 4.14 *For A and B on a B-spline surface, σ, A and B are* independent *of each other, if either $\mathcal{S}(A) \cap \mathcal{S}(B) = \emptyset$ or if $\mathcal{S}(A) \cap \mathcal{S}(B) = \mathbf{T}$ and $\hat{\eta}(A) = \rho_A$ and $\hat{\eta}(B) = \rho_B$ then for all $(i,j) \in T$, $\mu_{\rho_A}(i,j) \neq \mu_{\rho_B}(i,j)$.*

If the surface is not torn, then there is only one mesh, and the two points are independent of each other if the intersection of their spans is empty. If a surface is torn, then the spans can overlap, as long as each index pair in the intersection corresponds to a control point in a different mesh. Now we show that, in fact, nothing is missing from O^κ.

Definition 4.15 *A and B are* on opposite sides *of the tear γ_κ if the straight line (in parametric space) between A and B intersects the tear an odd number of times (where intersection with a tangent of the tear that is not an inflection counts as two intersections, and an intersection at a tear endpoint is one intersection).*

Even if two points are not on opposite sides of a tear by this definition, a complete classification, where every point is on one side of the tear or the other, is sometimes useful. The containment function, μ_c, is used to provide this kind of complete classification in Definition 4.12 of the torn B-spline surface.

Theorem 4.2 *If A and B are on opposite sides of the tear, γ_κ, and $\mathcal{S}(A)$ and $\mathcal{S}(B)$ are not part of a constraint region, then $\mathcal{S}(A)$ and $\mathcal{S}(B)$ are independent of each other.*

Proof. Assume A and B are independent of each other and on opposite sides of the tear γ_κ, and assume that $\mathcal{S}(A)$ and $\mathcal{S}(B)$ are not part of a constraint region. Let C be a point on γ_κ such that C lies on the line between A and B. Let $\hat{\eta}(A) = \rho_A$ and $\hat{\eta}(B) = \rho_B$. Suppose then that there exists an index pair, (i,j), such that $\mu_{\rho_A}(i,j) = \mu_{\rho_B}(i,j)$ and therefore $(i,j) \in \mathcal{S}(A) \cap \mathcal{S}(B)$. This implies that $(i,j) \notin O^\kappa$. Then by the Theorem 4.1, $(i,j) \in \mathcal{S}(C)$. However, by definition, all $(i,j) \in \mathcal{S}(\gamma_\kappa)$ are in O^κ unless they are part of a constraint region. If $(i,j) \in O^\kappa$ then $\mu_{\rho_A}(i,j) \neq \mu_{\rho_B}(i,j)$ since the A and B are in regions on opposite sides of the tear, and this is a contradiction. ∎

As indicated by the assumptions, Theorem 4.2 does not apply to the case when individual control points are removed from the overlap mesh in order to maintain smoothness in the regions surrounding the tear. Then there are fewer DOFs and points on opposite sides of the tear are, by design, not completely independent of each other. Results pertaining to constraint regions will be presented in Section 4.10.3.

These theorems show that the defined overlap meshes are both necessary and sufficient to provide the maximum flexibility around the discontinuity.

Unfortunately, the assumptions also state that the points A and B had to be on the opposite sides of a tear. If A and B are on the same side of a given tear, or more precisely, $\eta(A) = \eta(B)$, then they are dependent on each other as though the tear did not exist, since they are guaranteed to use the same set of control points by η. This implies that even if a tear should pass between the two points and double back before terminating, theoretically requiring independence between the points, this independence is not guaranteed. See Section 4.10.1 for a discussion of how to split tears so that η is defined appropriately.

4.8 Masking Function, μ

The masking function is the embodiment of the constraints within the data structure. As such, this function is dependent on the overall structure of the tears within the torn B-spline surface. Two aspects of the particular implementation discussed here need to be clear before proceeding. The first is that the tears are connected to the boundaries of the parametric region (or another tear) by adding invisible parametric line segments, called *extensions*, to the endpoints of the tears. These extensions partition the surface into disjoint parametric regions indicated by c. The containment function, η_c, is a classification function and embodies these disjoint regions; it is discussed further in Section 4.9. The second is the concept that the disjoint regions and the tears are related to one another by an ordering, called the *signature ordering*.

Definition 4.16 *A* valid signature ordering *for a torn B-spline surface is a (possibly partial) ordering of parametric regions and tears of a torn B-spline surface such that*

1. *every tear and every disjoint region are in the graph of the ordering,*
2. *every tear directly precedes one and only one region in the graph,*
3. *no region is preceded by more than one tear,*
4. *tears are not directly preceded by tears, and*
5. *regions are not directly preceded by regions.*

By this definition, a region may precede any number of tears (including none), but a region's direct precedence of more than one tear is discouraged for simplicity and ease of implementation.

Definition 4.17 *The* signature *for a region of a torn B-spline surface is defined by the portion of the signature ordering of the torn B-spline that precedes and includes the region.*

With these definitions, every distinct parametric region, c, has a unique signature. The implementation and other issues surrounding the signature ordering for a surface is discussed in Section 4.10.6.

Definition 4.18 *The* masking function*, $\mu_c(i,j) : \mathbb{Z}^2 \rightarrow \{0,\ldots,T\}$, determines if a particular coefficient, $O_{ij}^{(\kappa)}$, is used in parametric region, c. Let $(u_0,v_0) = \gamma_\kappa(t_{min})$, $(u_1,v_1) = \gamma_\kappa(t_{max})$ and $\mathcal{D} = \bigcup\{\mathcal{S}(u_i,v_i) | i = 0,1; (u_i,v_i)$* does not lie on another discontinuity (such as a surface boundary or another tear)$\}$. *Then $\mu_c(i,j) = \kappa$ if*

1. *there exists a point, (u,v), on γ_κ, such that $(i,j) \in \mathcal{S}(u,v) - \mathcal{D}$,*
2. *$O_{ij}^{(\kappa)}$ exists and κ is the last tear in the signature of c, and*
3. *$O_{ij}^{(\kappa)}$ has not been replaced by another DOF by being a member of the span of an extension (more details are in Section 4.10.7).*

There are special considerations for tears which intersect.

4.8.1 Intersecting Tears

A tear may intersect another tear in one of two ways: 1) end to end or 2) side to end. Other cases are not allowed by the definition of a tear curve. Where the endpoint of a tear lies is critical to determining the continuity of the surface near that point. If the endpoint of a tear lies on the edge of a discontinuity (either the boundary of the surface, or another tear), then the span of the endpoint may be used as part of the overlap mesh in its entirety because no continuity needs to be maintained at that point. The two discontinuities (in particular, the tears) are said to have a *parent-child* relationship at the point of intersection (see Figure 4.6). Since no modifications need to be made when processing the tear that the second tear abuts to, it is considered the *parent*. The abutting tear is considered the *child* since the computation of the additional coefficients depends on the intersection.

The end to end case is very similar in that a parent-child relationship is also determined. In this case, however, the particular relationship is not necessarily clear. In most cases, it makes no difference which tear is the parent and which is the child. The only difference seen by experimentation is that the determination of

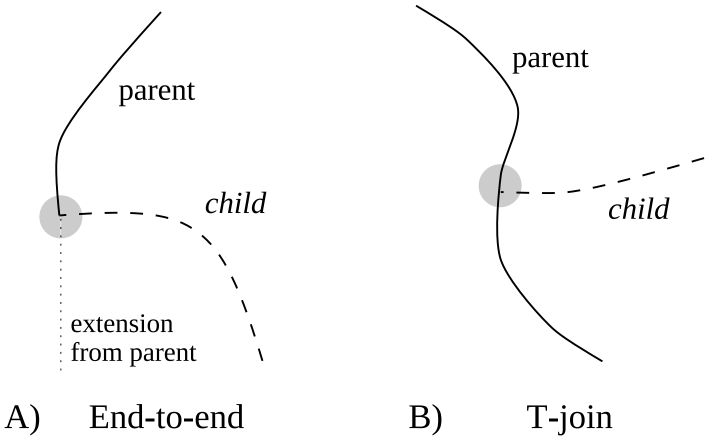

Figure 4.6. Example of parent-child relationships for two types of intersections.

the coefficients may be more complex for a particular relationship in some cases. The exact nature of this complexity relates to the composition of the control meshes for some regions. In particular, experiments have shown that fewer exceptions to the general composition rules are encountered when the parent tear precedes the child tear (see Section 4.10.6 for a discussion of precedence rules and ordering). The parent-child relationship pertains only to a given intersection. In particular, a set of tears may have a circular relationship when considering all endpoint intersections of all tears (see Figure 4.7).

These conditions provide the inherent constraints of the representation by enforcing continuity within patches containing the endpoints of the tear curves and providing additional DOFs to guarantee the independence of patches which are separated by the tear. For the definition and discussion of signatures, see Section 4.10.6. For a discussion of prevention and propagation, see Section 4.10.7.

4.9 Containment Function, η

Suppose that two points, A and B, are on opposite sides of a tear as shown in Figure 4.8. Then $\mathcal{S}(A) = \{(i,j)|i = 3,4,5,6; j = 2,3,4,5\}$ and $\mathcal{S}(B) = \{(i,j)|i = 3,4,5,6; j = 4,5,6,7\}$. Then $\mathcal{S}(A) \bigcap \mathcal{S}(B) = \{(i,j)|i = 3,4,5,6; j = 4,5\}$. If these

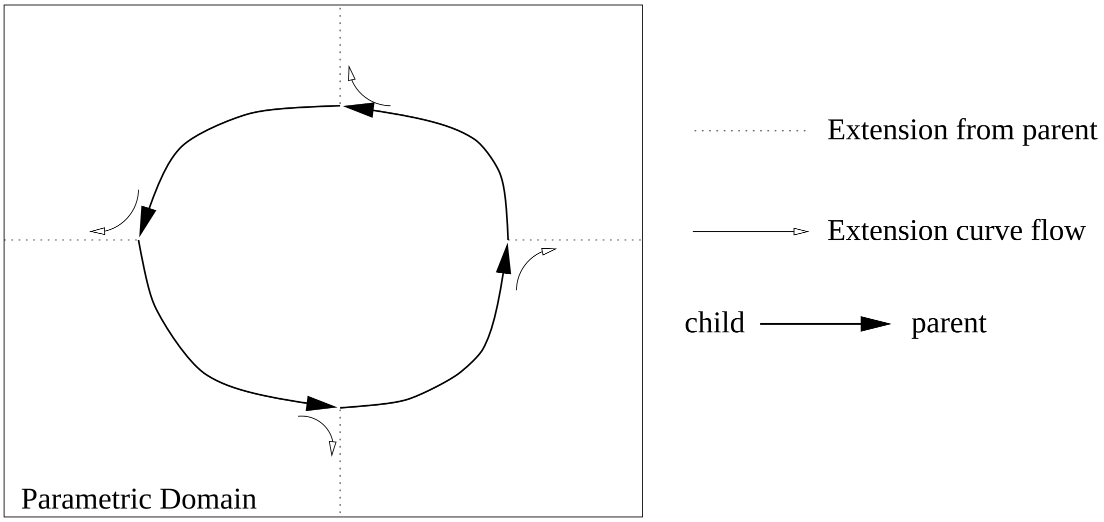

Figure 4.7. Example of circular parent-child relationships.

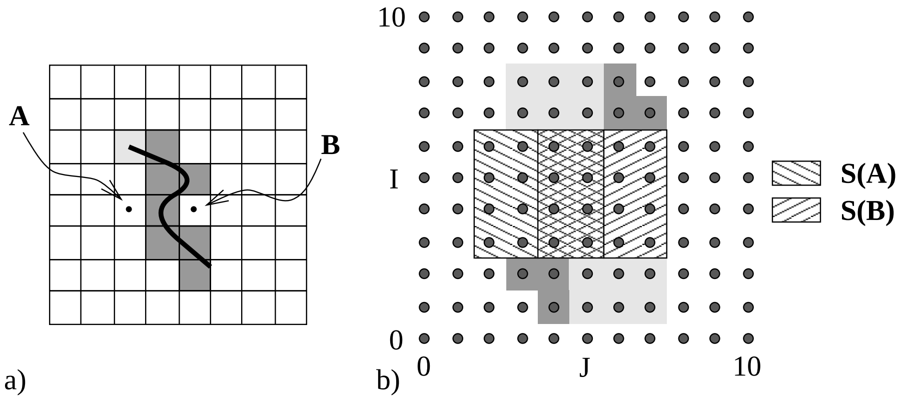

Figure 4.8. Illustration of overlapping spans for two given points. a) Parametric domain with points A and B on opposite sides. b) Control mesh indicating spans.

two points are to be independent, it is necessary to partition, or separate, the surface into regions which are evaluated with different sets of control points. In particular, control points with subscript pairs in the intersection of the spans must have a unique control point identity for each side. The containment function, η, classifies each parametric location with respect to each of these separated parametric regions. The containment function, η, can be constructed, for example, via a set of oriented boundary loops which are determined by extending tear curves to boundaries (see Figure 4.9). The parametric locations on the boundaries between parametric regions are defined in more than one parametric region so the classification must have more information. If the parametric location is on a tear, an *is_left* flag is used to disambiguate the results of this function. If the parametric location lies on an extension of the tear curve, an arbitrary choice can be made since the parametric regions must be continuous along the extension. If the parametric location is on a tear, this ambiguity is expected. Otherwise, the representation enforces smoothness in that region, so the choice is arbitrary.

Definition 4.19 *The* containment function $\eta_c(u,v) : \mathbb{R}^2 \rightarrow \{0,1\}$ *is the characteristic function of the restricted domain of the parametric region, c.*

$$\eta_c(u,v) = \begin{cases} 1 & \text{if } (u,v) \in domain(c) \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

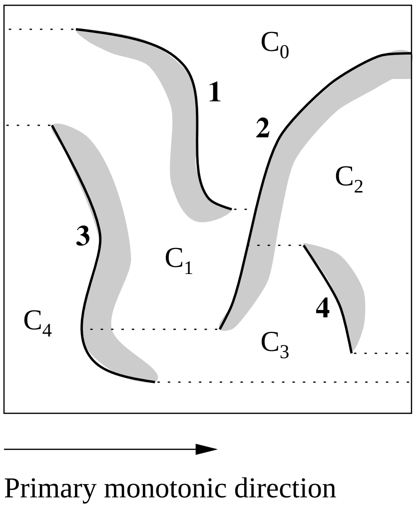

Figure 4.9. Tears in a surface with dashed extension curves.

$\sum_{c=0}^{T} \eta_c(u,v)$ represents the number of distinct parametric regions which contain a particular parametric location. In the interior, $\sum_{c=0}^{T} \eta_c(u,v) = 1$. Along interior boundaries, that is, tears and their extensions, $\sum_{c=0}^{T} \eta_c(u,v) = 2$. At points of intersection of the interior boundaries, the sum can be higher.

4.10 Implementation

In Section 4.7 it was shown that the additional degrees of freedom created by the introduction of a tear into the torn B-spline surface are both necessary and sufficient to provide the maximum flexibility allowed by the discontinuity. Determining the mask values μ for each parametric region is probably the most challenging task in implementing the torn B-splines. Intuitively, each additional coefficient created by the discontinuities in the surface must be used by a parametric region to differentiate it from its adjacent parametric regions across the discontinuity. This is accomplished by assigning to each parametric region a unique combination of additional DOFs from the tears in the structure.

One procedure which determines the parametric regions and their unique combinations of DOFs can be outlined as follows:

1. Split tears into monotonic segments.

2. Compute $O^{(\kappa)}$.

3. Determine necessary extension directions and extend curves.

4. Separate the domain into parametric regions, c_ι, i.e., define η_c.

5. Order parametric regions and tears.

6. Construct $\mu_c(i,j)$.

4.10.1 Splitting Tears

Tear curves are split into monotonic sections with respect to the parametric domain for two primary reasons. Foremost, the monotonicity provides a framework within which parametric regions and tears can be ordered without cycles. A natural ordering can then be derived from adjacency and parametric value information. This is important for determining a unique set of control points for each parametric region. The second reason can be most clearly seen by considering some examples. In Figure 4.10, the tear curve enters and leaves the same patch twice (known as "doubling back"). Intuitively, this should cause the patch to be split into three independent patches (I, II, and III in Figure 4.10). However, since the two outside patches are on the same side of the tear, they will belong to the same parametric region and therefore use the same set of control points. That is, the two outside sections, I and III, will not be independent as expected. A related example is in Figure 4.11. In this figure, the tear curve "spirals" back into the same patch again, causing even more confusion. In Figure 4.10 it is at least clear which region each section of the patch belongs to, even though they are dependent. In Figure 4.11, the center section of the patch in question belongs to the regions on both sides of the curve. Unless the two parts of the tear curve which pass through the patch are differentiated, the description of the surface is ambiguous. In both of these examples, it is not necessary for the curves to pass through the same patches to cause these problems. They only need to pass through patches which are dependent on the same coefficients. It is easy to see that in most cases, if a curve spirals or

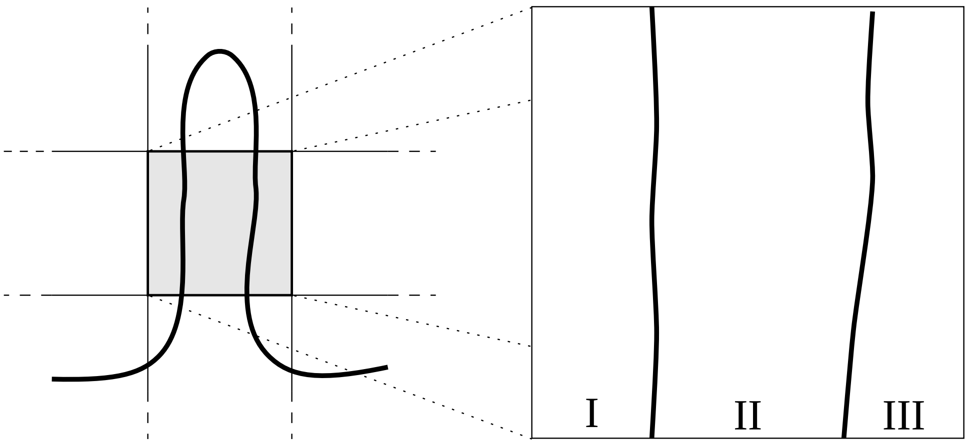

Figure 4.10. Example of a curve that "doubles back."

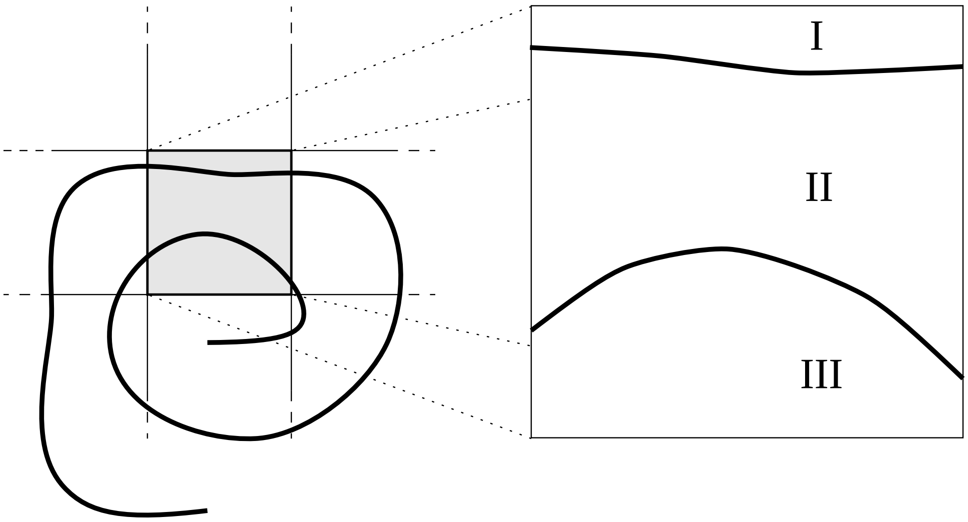

Figure 4.11. Example of a curve that spirals.

doubles back, that this type of problem is likely to occur.

Spirals are eliminated completely by splitting new tear curves into monotonic sections in any given primary direction since to spiral the curve must have a minimum or maximum relative to any given direction. However, isolating monotonic sections in a spiral may result in a given curve section doubling back (see I, II, and III, in Figure 4.12). Unfortunately, the doubling back case is more difficult to correct. Splitting the curve into monotonic sections with respect to both primary parametric directions will solve this problem, but splitting curves this way in all cases is not necessary. It would be the most effective if the problem cases could be identified and split only if necessary. Since identifying these problems (especially the doubling back case) is nontrivial, they will be left for future work.

Splitting a tear results in two independent curves whose endpoints intersect. If endpoints intersect by accident, it is usually difficult to detect, but in this case the exact nature of the intersection is known. The parent-child relationship (introduced in Section 4.8.1) can be assigned arbitrarily by ordering the curve segments

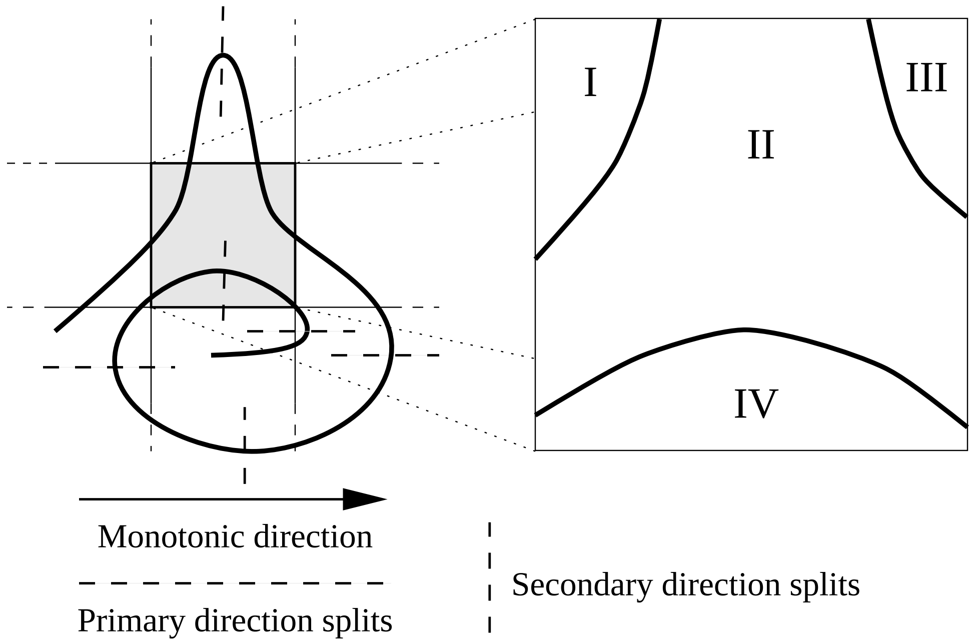

Figure 4.12. Example of a curve that spirals and "doubles back" and the monotonic splits that may be required.

according to beginning parametric value. The parent's endpoint in question is extended in the monotonic direction whereas the child's endpoint is not extended (see Section 4.10.3).

4.10.2 Computation of $O^{(\kappa)}$

Computation of $O^{(\kappa)}$ proceeds directly from the maximal independence algorithm presented in section 4.7.1.

4.10.3 Determination of Extension Directions

The tears are extended from their endpoints in the parametric domain out to another boundary, either an actual boundary of the surface, or another tear or extension. The span of the extension is considered a continuity constraint region, since an extension bounds two or more surfaces whose control points in the span of the extension must be the same. Figure 4.9 shows an example of a surface with a set of tears and their extensions. The regions in Figure 4.9 labeled C_i are the parametric regions separated by the extended tears. The wise choice of the extension direction is essential to a maximally independent surface.

The extension directions are crucial because they influence the ordering of the parametric regions and tears and affect the ease of maintaining the boundary conditions within the structure. Since the extensions are added by the structure, they need to be as unobtrusive as possible. In light of the discussions in the previous section on the importance of monotonicity, it would seem that the best extension would maintains the monotonicity of the tear in the primary direction. Particularly, the span of the extension must avoid intersecting the overlap mesh of the tear. Since the span of the extension is shared by regions on both sides of the tear, automatically maintaining continuity across the extension would be impossible. This is now developed formally.

Definition 4.20 *An extension, ϵ, of γ_κ is* valid *if $(i,j) \notin \mathcal{S}(\epsilon)$ for all $(i,j) \in O^\kappa$.*

To see why a valid extension is required, suppose that span of the extension curve contains a pair (i,j) that is also contained in the set of control points of O^κ.

Choose a point, x, on the extension curve such that $(i, j) \in \mathcal{S}(x)$ (see Figure 4.13). Because of the continuity of B-splines, there exists two points, $\hat{x}_1$ and $\hat{x}_2$, in the neighborhood of x such that $\hat{x}_1$ is on one side of the extension curve and $\hat{x}_2$ is on the other side. Given that the region on one side of the extension does not use O^κ and the region on the other side does, then these two points are not dependent on the same control points and the regions are not the same along the extension curve at x even though they are required to be.

Even with this criteria, there are generally several options, two of the most obvious are given here.

1. Extend the tear by continuing the tear in the tangent direction (in parametric space) of the tear at the endpoint. This approach has the advantage of being easy to understand conceptually, and it maintains the continuity of the tear at the endpoint.

2. Extend the tear by an isoparametrically straight segment in the primary monotonic direction, away from the tear in the direction closest to the tangent at the endpoint. The advantages of this approach are that the signature ordering tends to remain the same independent of the order in which tears

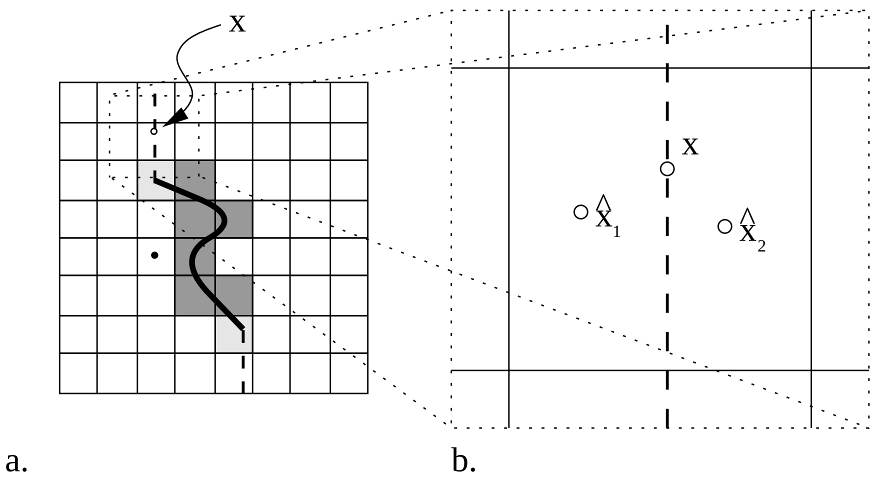

Figure 4.13. Points in the neighborhood of x on either side of the extension.

are extended, and it is generally easier to deal with computationally.

For computational purposes, the second option was chosen for implementation in this thesis. For tears that have been split into monotonic sections, the tangent of the tear at the split point is perpendicular to the monotonic direction, so extensions are made in the direction opposite the normal of the tear relative to parametric space. In the case of a parametrically straight tear, where the normal is not well-defined, the choice is arbitrary. If convenient, the tear may be extended so that the orientation of the extended tear relative to the primary monotonic direction will match that of the other tears.

Another special case occurs when two extensions "run into each other." This happens when the extension directions are opposites of each other and the end-points have the same position perpendicular to the monotonic direction (as in Figure 4.14(A)). The are several options in this case.

1. Choose one curve and extend it slightly perpendicular to the monotonic direction and then proceed as before (as in Figure 4.14(B)).

2. Choose another direction for one of the tears (as in Figure 4.14(C)).

The first option is also probably the best option. This option has the advantage of keeping the ordering characteristics provided by the monotonicity of the tears. The second option has the advantage that a certain set of directions can be preset so that the monotonicity can be maintained and the various choices can be permuted until a solution is found.

Since these extensions are critical to the separation of regions and ultimately to the composition of coefficients for those regions, a solution must be guaranteed if an option is to be viable. A viable solution has the following characteristics:

1. all extensions of tear curves satisfy the requirements of avoiding the overlap mesh of their corresponding tears,

2. no extension may terminate at the endpoint of another tear.

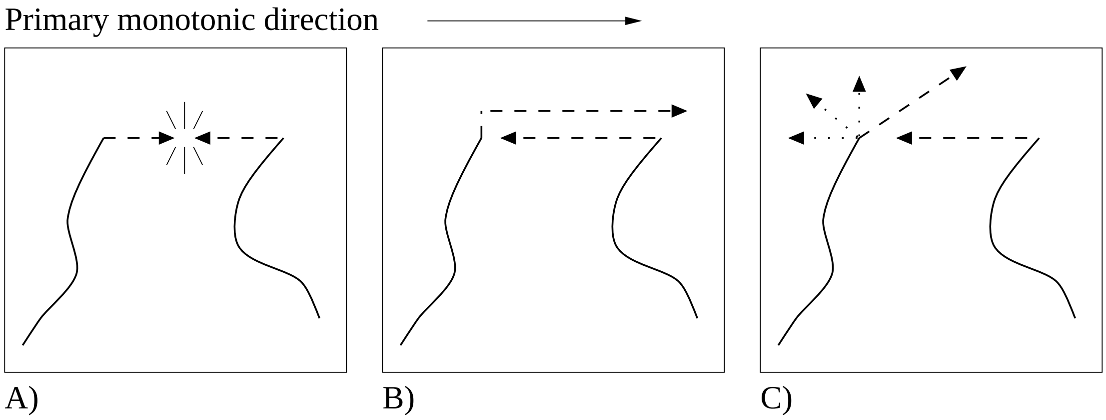

Figure 4.14. Extensions that "run into each other." (A) The problem. (B) Solution 1: Right angle sidestep. (C) Solution 2: Alternate directions.

It is easy to see that the first option given above will always provide a solution. Since there are a finite number of possible extension curves (two for each tear curve) and an infinite number of extension distances, there will always be a distance to extend the curve before turning in the monotonic direction so as to avoid running into any other extension.

The viability of second option is a bit more difficult to show. In order for two extensions to "run into each other," they must be aligned exactly and moving in opposite directions. Further, no other tear endpoint lies on this "line of sight" between the two endpoints (if so, the two would not run into each other). Suppose we label these two extensions as a *conflicting pair*. Let $n(< 1)$ be the number of preset extension directions (given as offsets to the tangent of the curve at the endpoint). Suppose, then, that there exists at least one conflicting pair and that for both extensions, all other possible directions would result in a conflicting pair. Taking it one step further, suppose that the one of the extensions switches directions in response to the conflict. In order to continue having a problem, the new conflicting pair must have potential conflicts in all directions. In the limit, there must be $(2T)^n$ extensions causing conflict, where T is the number of tears, since each extension has a conflicting extension in each direction. But if $n > 1$, this is impossible since there are only $2T$ extensions. Therefore, as long as there are 2 or more possible extension directions, the second option will provide a solution.

Unfortunately, the second option may require finding a solution among $(2T)^n$ possibilities, so heuristics should be used to govern the selection process. Fortunately, experimentation has shown that although the possibility of a conflict occurring is high, especially in artificial situations, a simple set of rules will result in a solution in $O(T)$ time. Following is a brief list of one rule set that is effective.

4.10.4 Initial Extension Rules

Given a curve, determine the initial guess of the extension direction for the extension at the beginning of the curve by the following rules. For the sake of terminology, the tangent of the curve at its beginning points in the direction of curve. The inverse of this tangent points away from the curve. However, the tangent of the curve at the end of the curve points away from the curve and the inverse tangent points toward the curve. All evaluations are relative to the parametric space of the surface.

1. Determine which of the primary monotonic direction or its inverse is closest to the inverse tangent direction at the beginning of the tear. Use this direction if the tangent at the beginning of the tear is not perpendicular to the primary direction.

2. If the tangent at the beginning of the tear is perpendicular to the primary monotonic direction, determine the direction closest to the inverse of the tangent at the end of the tear. Use this direction if the tangent at the end of the tear is not perpendicular to the primary direction.

3. If both ends of the tear have tangents perpendicular to the primary monotonic direction, use the extension direction closest to the vector from the end of the tear to the beginning of the tear, provided this vector is not perpendicular to the primary monotonic direction.

4. If all of the above fail, use the secondary monotonic direction closest to the inverse of the tangent at the beginning of the curve. This is guaranteed (except for rare degenerate cases) to be nonzero if all of the above fail.

Once an initial guess is made, any enumerative technique can be used to find a solution for which there are no conflicts.

The removal of conflicting pairs is necessary in order to produce exactly $T+1$ separate parametric regions. If T tears are in the surface, there is the potential for $T+1$ unique coefficients for a particular (i,j) pair in the torn surface. There must be a unique parametric region for each of these potential coefficients in order to provide the maximum flexibility for each of the tear curves. If a conflicting pair persists, the number of separate parametric regions is reduced by 1.

After the directions have been determined, the curves are extended and information regarding intersections is compiled for each tear.

4.10.5 Parametric Regions, η_c

The current implementation uses trimming loops to separate the domain into parametric regions although other techniques such as a quadtrees may be just as applicable. Information about where the extended tear curves intersect the boundaries and other tears is used to compile a set of boundary loops. One notable side effect of this process is that the particular curve segments of the tear curves which make up the boundary for a given parametric region can be cached. This information is used to determine the signature ordering of the parametric regions and tears (see Section 4.10.6).

4.10.6 Ordering Parametric Regions and Tears

Finally, the piecewise functions, $\mu_c(i,j)$, determine the composition of coefficients from the overlap meshes and original mesh in each of the parametric regions. We refer to the ordering which determines these compositions as a *signature* ordering (defined in Section 4.8). The signature of each parametric region must be unique. The signature ordering is an alternating, possibly partial, precedence relation, beginning with a single root parametric region. The ordering then alternates after this: tear, region, tear, region, etc. Every tear and parametric region are required to be part of this precedence relation. From this relation, the signature ordering for each of the parametric regions is the portion of the precedence graph which precedes the given parametric region (defined formally in Section 4.8). The precedence graph is denoted by an ordered set. For example, suppose a precedence relation for a given torn B-spline surface is $c_I < \gamma_A < c_{II} < \gamma_B < c_{III}$, where c_i are the parametric regions and γ_i are the tear curves. The root of the graph is c_I and its signature ordering is denoted by the empty set, $\{\}$, since nothing precedes it. The signature ordering for c_{II} is denoted $\{c_I\gamma_A\}$. In a similar fashion, the signature ordering for c_{III} is denoted $\{c_I\gamma_A c_{II}\gamma_B\}$. Although every tear and region must be included in the signature for the surface, at certain points the relation is only partially specified. In this case the notation used is illustrated in the following example: if $c_I < \gamma_A < c_{III}$ and $c_{II} < \gamma_B < c_{III}$ are the partial orderings that are known for the surface, then $\{[\{c_I\gamma_A\}, \{c_{II}\gamma_B\}]c_{III}\}$ denotes the precedence relation for the surface. The rules of precedence are given in Section 4.10.6.2 below.

A valid signature ordering does not require that an ordered relationship exists between all combinations of tears and regions. It is possible for the relation to have branches (see cases 2a and 2b, below), in which case a precedence relationship is not defined between members of the branches. However, branching is allowed only in some cases (for examples of inappropriate branching, see cases 1a and 1b, below). It is usually desirable for the precedence relation to be totally ordered, even if the signature ordering is valid. In the case of branches, either a specific method is used to create a total ordering or an arbitrary precedence is assigned for convenience. There are two primary reasons to create a total ordering. First, in some cases it is necessary to prevent DOFs from being dropped. Second, if it is not necessary, then assigning an arbitrary order will not affect the outcome. The reasoning behind this is that if a particular ordering affected the outcome, where the outcome is the interdependence between two parametric regions, then either a precedence relationship can be determined, or it is required (as in the branching case, above). The essential requirement is that the signatures be different: this is what is required to provide the maximum independence between regions. As long as the signatures follow the rules below, any set of unique signatures will result in the same amount of freedom. This can be easily shown by observing that if the signatures are unique, then the difference in composition between adjacent surfaces is the addition of the DOFs associated with the tear which forms the boundary. Therefore, completing the ordering is a good idea regardless of the particular situation. Determining an order with the fewest complications is best accomplished by reorienting the tears so they all travel in approximately the same direction. Then the ordering can easily be determined. For the purposes of this discussion, tears refer to the actual tear combined with its extensions.

4.10.6.1 Reorientation of Extended Tears

Once we have determined the necessary extensions for each of the tear curves, the extended tear curves can be reoriented so that all of the tears are oriented in the same direction within the domain of the surface. If v is the primary monotonic direction, then the extended curves, except for the few rare cases, can be oriented so that the beginning of the curve has the smallest v value. In the rare case in which the tear curve was extended in the secondary monotonic direction, the reorientation is relative to the secondary monotonic direction (in this case, u). After reorientation, the curve's left side is the region to the left of the curve when looking from the beginning to the end.

4.10.6.2 The Precedence Relation

The precedence relation is determined by the following rules.

Rule 1: A parametric region, c_I, precedes a tear, γ_A, $c_I < \gamma_A$, if any part of γ_A's left side lies on an edge of c_I.

Rule 2: A tear, γ_A, preceeds a parametric region, c_I, $\gamma_A < c_I$, if γ_A's right side lies on an edge of c_I.

Rule 3: (Transitivity) If $c_I < \gamma_A$ and $\gamma_A < c_{II}$, then $c_I < \gamma_A < c_{II}$.

Rule 4: No tear may be in immediate predecessor of more than one parametric region.

These rules create a partial ordering (possibly with branches) of alternating parametric regions and tears within the surface. Additional rules used to create a total ordering are given below. When all parametric regions have a unique signature ordering, the signatures are considered to be *valid*.

There are several tear configurations which result in branching of the signature ordering. To provide a valid signature ordering, additional relational pairs may be necessary. The cases in question are:

1a. Two tears intersect the left parametric boundary where the lower tear is oriented toward the boundary and the upper tear is oriented away from the boundary (Figure 4.15).

1b. Two tears intersect the right side of the same tear within the surface where the lower tear is oriented toward the boundary and the upper tear is oriented away from the boundary (Figure 4.15).

2a. Two tears intersect the right parametric boundary where the lower tear is oriented away from the boundary and the upper tear is oriented toward the boundary (Figure 4.16).

2b. Two tears intersect the left side of the same tear within the surface where the lower tear is oriented away from the boundary and the upper tear is oriented toward the boundary (Figure 4.16).

Cases 1a and 1b are similar with the parametric boundary, b, and the larger tear, γ_C, playing the same role. In both cases, the parametric regions, c_I and c_{III}, are not ordered with respect to each other. From the rules given previously, $c_I < \gamma_A$, $\gamma_A < c_{II}$, $c_{III} < \gamma_B$ and $\gamma_B < c_{II}$, and in Figure 4.15(1a), $c_{IV} < \gamma_C$, $\gamma_C < c_I$, $\gamma_C < c_{II}$ and $\gamma_C < c_{III}$ in addition (see Figure 4.15(1b)). The signatures with this partial ordering are

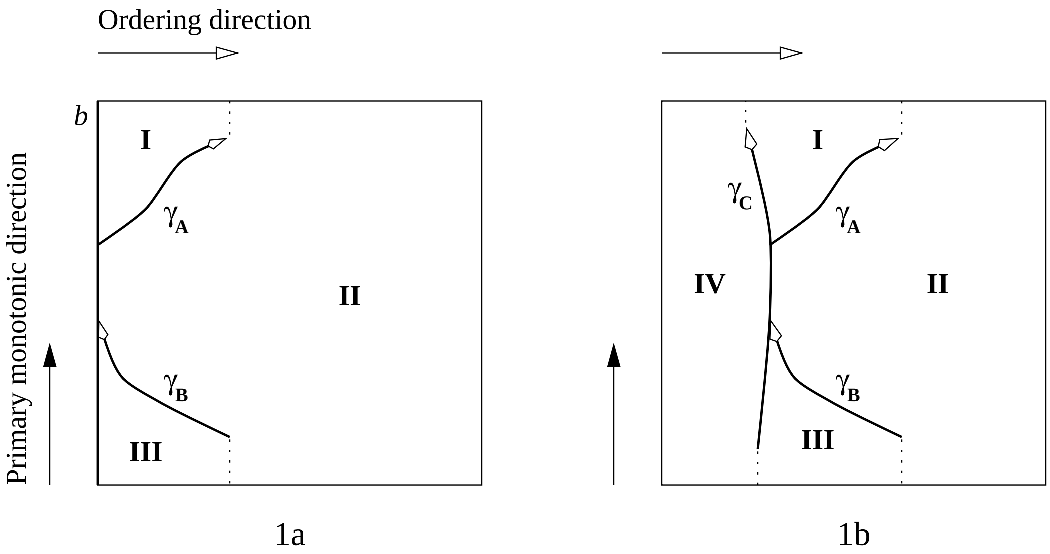

Figure 4.15. Common boundary signature conflicts.

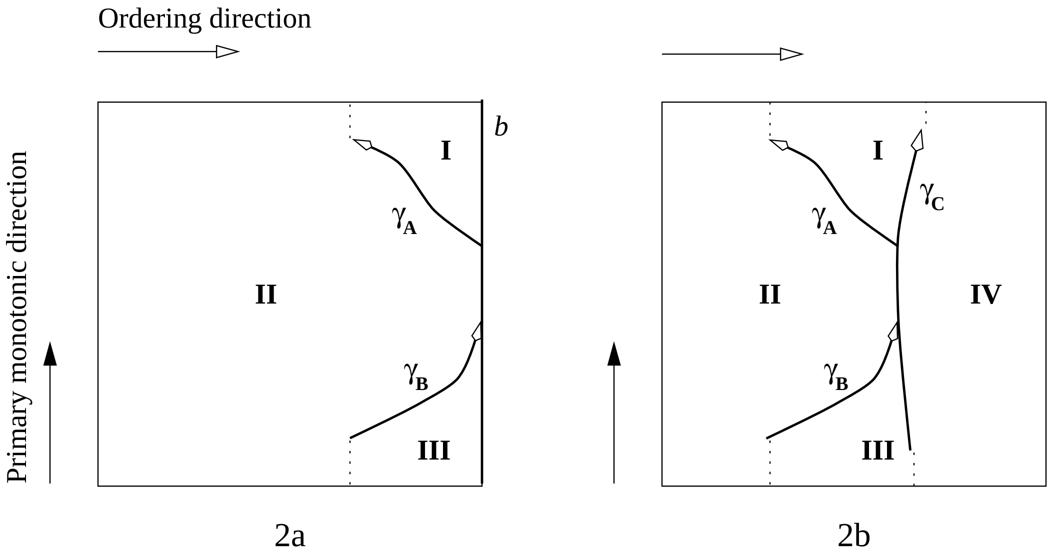

Figure 4.16. Branching of signatures.