Dynamic PDE Surfaces with Flexible and General Geometric Constraints

Haixia Du **Hong Qin**
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794 – 4400, USA
$\{$dhaixia, qin$\}$@cs.sunysb.edu

Abstract

PDE surfaces, whose behavior is governed by Partial Differential Equations (PDEs), have demonstrated many modeling advantages in surface blending, free-form surface modeling, and surface's aesthetic or functional specifications. Although PDE surfaces can potentially unify geometric attributes and functional constraints for surface design, current PDE-based techniques exhibit certain difficulties such as the restrained topological structure of modeled objects and the lack of interactive editing functionalities. We propose an integrated approach and develop a set of algorithms that augment conventional PDE surfaces with material properties and dynamic behavior. In this paper, we incorporate PDE surfaces into the powerful physics-based framework, aiming to realize the full potential of the PDE methodology. We have implemented a prototype software environment that can offer users a wide array of PDE surfaces with flexible topology (through trimming and joining operations) as well as generalized boundary constraints. Using our system, designers can dynamically manipulate PDE surfaces at arbitrary location with applied forces. Our sculpting toolkits allow users to interactively modify arbitrary point, curve span, and/or region of interest throughout the entire PDE surface in an intuitive and predictable way. To achieve real-time sculpting, we employ several simple, yet efficient numerical techniques such as the finite-difference discretization, the multi-grid subdivision, and the FEM approximation. Our experiments demonstrate many attractive advantages of physics-based PDE formulation such as intuitive control, real-time feedback, and usability to both professional and non-expert users.

Keywords: *PDE Surfaces, Geometric Modeling, CAD/CAM, Dynamic Modeling, Geometric Constraints, Interactive Techniques.*

1. Introduction

Surface modeling techniques are vital to many visual computing fields such as interactive graphics, CAD/CAM, animation, and virtual environments. Although a wide range of free-form splines have been developed during the last several decades [8, 11, 14, 13], traditional spline-based techniques can be difficult, time-consuming, less natural, and counter-intuitive, primarily because free-form splines are associated with tedious and indirect shape manipulation through time-consuming operations on a large number of (oftentimes irregular) control vertices, non-unity weights, and/or non-uniform knots. In addition, strong mathematical sophistication is necessary for users.

By contrast, PDE surfaces have recently emerged as a powerful modeling technique and started to gain popularity and strength for surface modeling and design. PDE surfaces permit geometric objects to be defined and governed by a set of differential equations. In comparison with traditional control-point-based techniques, PDE surfaces offer many advantages:

- Natural physical processes are frequently characterized by PDEs. In principle PDE surfaces can be controlled by physical laws, so they are natural and much closer to the real world. PDEs are potentially ideal candidates for both design and analysis purposes.

- The formulation of differential equations is well-conditioned and technically sound. Smooth surfaces with high-order continuity requirements can be readily defined through the use of complicated PDEs.

- Smooth surfaces that minimize certain energy functionals oftentimes are associated with differential equations, so optimization techniques can be integrated with PDE surfaces.

- Many powerful numerical techniques to solve PDEs are commercially available. Parallel algorithms can be deployed for large-scale problems in industrial settings.

- Users can easily understand the underlying physical processes associated with PDEs, therefore, intuitive and natural control is possible through the modification of physical parameters.
- PDE surfaces can potentially unify both geometric and physical aspects. They are invaluable throughout the entire modeling, design, analysis, and manufacturing processes. Various heterogeneous requirements can be enforced and satisfied simultaneously.

Despite the rapid advances and modeling successes of PDE surfaces, they demand a lot of novel interactive techniques in order to realize their full potential. Typical modeling difficulties associated with PDE surfaces include:

- The prior work on PDE surfaces mainly concentrates on elliptic PDEs and is lack of interactive techniques for direct shape manipulation.
- Besides simple geometric conditions along PDE surface boundaries, as well as the manual editing of PDE coefficients, there is a lack of formal mechanism for the direct manipulation of PDE surfaces.
- Traditional elliptic PDE surfaces only result from Hermite-like boundary conditions (i.e., boundary curves and their corresponding derivatives up to order n). More flexible and general boundary constraints are not yet addressed.
- Conventional PDE techniques are unable to support localized geometric operations. Global control is less intuitive to manipulate.

To ameliorate it, we [9] recently proposed an interactive methodology and developed novel modeling techniques that can facilitate the direct manipulation and interactive sculpting of PDE surfaces. Our algorithms and design framework are founded on the integrated principle of differential equations and physics-based modeling. To further promote the applicability of PDE surfaces in interactive graphics, CAD/CAM, and virtual engineering, we shall forge ahead towards the realization of the full modeling potential associated with dynamic PDE surfaces. In this paper, we extend both the geometric coverage and topological variation of PDE surfaces. Our new system provides users a set of more powerful sculpting tools than previously-developed point editing capabilities. These toolkits allow PDE surfaces to be defined through the use of general, flexible boundary constraints. Complicated geometry and diverse types of topology for PDE surfaces are readily available in our modeling environment. Other typical design tools in our environment include trimming, merging, the manipulation of a set of isoparametric curves and/or arbitrary curve networks, the editing of any user-specified sub-surface, etc. Using our system, users are able to enforce both physical requirements and geometric criteria on PDE surfaces simultaneously with ease.

The remainder of the paper is structured as follows. Section 2 reviews the prior work of PDE surfaces and physics-based models. In Section 3, we detail the PDE formulation and present our integrated approach. Section 4 discusses novel techniques of directly manipulating PDE surfaces with generalized boundary constraints and flexible topology. We outline the system implementation and document our experimental results in Section 5. Section 6 concludes the paper.

2. Prior Work

In 1989 Bloor and Wilson [1] pioneered a modeling method—PDE surfaces—that defines a smooth surface as a solution of *elliptic* PDEs. Since their initial application on surface blending, PDE surfaces have broadened their uses in surface description, solid modeling, and B-spline approximation in recent years. In principle, the PDE-based method has the advantage that most of the information defining a surface comes from its boundary curves. This permits a surface to be generated and controlled by very few parameters such as boundary-value conditions and global coefficients associated with an elliptic PDE. In addition, this PDE technique can be used to generate piecewise free-form surfaces [3]. By varying boundary conditions and control parameters in PDEs, designers can obtain various surface shapes. Furthermore, Lowe, Bloor and Wilson [12] presented a method with which certain engineering design criteria such as functional constraints can be incorporated into the geometric design of PDE surfaces. Therefore, it may simultaneously introduce geometric constraints, aesthetic criteria, and physical and engineering restrictions into design process. Additionally, Bloor and Wilson [2] have developed an algorithm that approximates PDE surfaces using standard B-splines. This work intends to demonstrate that PDE surfaces are virtually compatible with other matured and well established spline-based techniques for surface design, hence PDE surfaces can be readily incorporated into existing commercial design systems. Later on, in 1993 PDE solids were formulated in terms of parametric boundary surfaces [4], which further expands the geometric coverage of the PDE methodology. For certain simple boundary conditions, some elliptic PDEs can be solved analytically, i.e., PDE surfaces in these cases have a closed-form formulation that frequently involves functions of Fourier series. However, for general boundary conditions, a PDE solution will have to be sought numerically instead. Recently, Bloor and Wilson [5] derived a set of approximate analytic solutions for PDEs subject to general boundary conditions. The approximate solution can be made to approach the true solution up to any degree of accuracy. Their generic solutions can be decomposed into a finite sum of Fourier functions that satisfy PDEs with an additional 'corrector' term that satisfies boundary conditions. In 1999, Ugail *et al.* [19] developed some techniques for interactively defining and changing boundary conditions in order to construct PDE surfaces.

Nonetheless, the above-mentioned techniques can only afford users indirect and non-intuitive shape manipulation on PDE surfaces. Physics-based modeling, in contrast, offers users a means to overcome the drawback of indirect design mechanism associated with PDE surfaces. It is possible to unify the physics-based modeling methodology with the PDE approach, mainly because that the dynamic behavior of physics-based models is also controlled by certain differential equations (e.g., Lagrangian equations of motion). Hence, physics-based modeling augments (rather than replaces) the existing PDE methodology, offering extra advantages for shape modeling. Terzopoulos and Fleischer [17] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [6] developed an interesting prototype system for interactive free-form design based on the finite-element optimization of energy functions proposed by Terzopoulos and Fleischer [17]. Terzopoulos and Qin [15, 18] formulated Dynamic NURBS (D-NURBS), a novel model for interactive sculpting. Ye, Jackson and Patrikalakis [20] incorporated certain functional constraints into the design process of geometric shape. Dachille *et al.* [7] presented a haptic approach for the direct manipulation of physics-based B-spline surfaces. Since the majority of physical phenomena can be characterized by PDEs, it is necessary to bridge the gap between geometric PDE surfaces and physics-based modeling approaches towards the realization of the full potential of PDE surfaces.

3. PDE Formulation

This section formulates PDE surfaces, and outlines properties of the unified principle of PDE surfaces and physics-based modeling.

3.1. Elliptic PDEs

Throughout this paper, we focus on the fourth-order elliptic PDE (which is a generalized version from [1]):

$$(\frac{\partial^2}{\partial u^2} + a^2(u,v)\frac{\partial^2}{\partial v^2})^2\mathbf{X}(u,v) = \mathbf{0} \tag{1}$$

where u and v are parametric coordinates over 2-space, $a(u,v)$ is a smoothing function of u and v that controls the behavior of PDE surfaces locally, and $\mathbf{X}(u,v) = \begin{bmatrix} x(u,v) & y(u,v) & z(u,v) \end{bmatrix}^\top$ define the PDE surface coordinates in 3-space. Note that, in [1] the control parameter is a constant a. To offer users more flexibility for interactive manipulation, we replace this constant parameter using an arbitrary function of u and v, which can be defined by users. Because $a(u,v)$ varies across $\mathbf{X}(u,v)$, local control on PDE surfaces can be achieved. Moreover, although our system is focused on this particular elliptic PDE, our mathematical derivation and its associated numerical techniques can be readily generalized to other PDEs. Because (1) is a fourth-order PDE, at least four boundary conditions are required in order to derive a unique solution. We further assume that our PDE surfaces are geometrically either closed or open along the two parametric directions (i.e., u and v), respectively. Therefore, our PDE surfaces may be topologically flexible, yielding diverse types of surfaces that are equivalent to four-sided open patches, spheres, cylinders, tori, etc. Reparameterization process can be conducted without changing the geometry of PDE surfaces if either u or v belongs to $[a,b]$. So, we restrain u and v to vary between 0 and 1. Various boundary conditions can be imposed. To simplify our implementation, we classify PDE surfaces into three types: (1) open along both u and v directions, (2) open along u-direction and closed along v-direction, and (3) closed along both directions. The four boundary conditions in our previous work comprise two curves which define a pair of the curved surface boundaries at the opposite side along u-direction and a pair of their associated derivative curves defining gradient information across the two curved boundaries. They are of the following form:

$$\begin{array}{l} \mathbf{X}(0,v) = \mathbf{c}_0(v), \mathbf{X}(1,v) = \mathbf{c}_1(v), \\ \frac{\partial \mathbf{X}}{\partial u}(0,v) = \mathbf{d}_0(v), \frac{\partial \mathbf{X}}{\partial u}(1,v) = \mathbf{d}_1(v). \end{array} \tag{2}$$

In this paper, we enhance PDE surfaces and generalize their boundary conditions to a curve network. This can facilitate the cross-sectional design of PDE surfaces from a set of (non-isoparametric) curves. For instance, consider the design technique of Gordon surfaces and Coons patches, our generalized boundary constraints can have the following form:

$$\mathbf{X}(u_i,v) = \mathbf{f}_i(v), \mathbf{X}(u,v_j) = \mathbf{g}_j(u), \tag{3}$$

where $0 \le u_i \le 1$ and $0 \le v_j \le 1$, and $\mathbf{f}_i(v)$ and $\mathbf{g}_i(u)$ are isoparametric curves. Moreover, a set of non-isoparametric curves can be easily incorporated into our formulation and algorithm.

3.2. Numerical Simulation

Previous work on PDE surfaces mainly seeks closed-form analytic solutions (e.g., Fourier series functions) in order to exploit many attractive properties of explicit formulations for surface modeling. Given arbitrary boundary conditions, however, we must resort to numerical techniques that guarantee solutions for PDE surfaces of flexible topology. Numerical algorithms also facilitate the material modeling of anisotropic distribution and its realistic physical simulation, where there exist no closed-form analytic solutions for dynamic PDE surfaces. Among many matured techniques, we employ finite-difference approach in our framework in the interest of real-time performance.

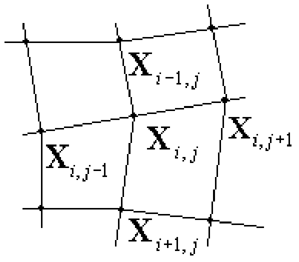

Figure 1. The point discretization of a continuous surface.

The finite-difference method transforms a continuous PDE to a system of algebraic equations by replacing all partial derivatives in differential equations with their discretized approximation. The system of algebraic equations can then be solved numerically either through a direct procedure or an iterative process to obtain an approximate solution of the continuous PDE. We use the central-difference approximation: $f'(x) = (f(x+h) - f(x-h))/2h$, $f''(x) = [f(x+h)-2f(x)+f(x-h)]/h^2$, where h denotes the spatial step. It is trivial to generalize the computation of univariate derivatives to all partial derivatives on bivariate surface geometry, by dividing the $[u, v]$ domain into m and n discretized points (see Fig. 1), respectively. Now, (1) can be rewritten as:

$$\mathbf{AX} = \mathbf{b}, \tag{4}$$

where $\mathbf{A}$ is a discretized differential operator in the $(mn) \times (mn)$ matrix form (note that, $\mathbf{A}$ is also controlled by the blending function $a(u, v)$), and

$$\mathbf{X} = \left[\begin{array}{cccc} \mathbf{X}_{1,1} & \mathbf{X}_{1,2} & \cdots & \mathbf{X}_{m,n} \end{array} \right]^\top,$$

$$\mathbf{b} = \left[\begin{array}{cccc} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_{m\times n} \end{array} \right]^\top.$$

Different topological types are available. First, the surface can be closed along one parameter direction (e.g., v), in which case the points on $v = 0$ are the same as those on $v = 1$. The central-difference scheme suffices for the computation of partial derivatives with respect to v. Second, the PDE surface is open along both u and v directions. In this case, the computation of partial derivatives on two boundary curves requires special care, forward or backward differences shall be utilized along the open boundary curves instead. Third, the PDE surface is closed along both directions, and the central-difference can be applied anywhere across the surface geometry. Boundary constraints determine all the point coordinates lying on the user-specified curves (e.g., $\mathbf{X}_{1,j}$ and $\mathbf{X}_{m,j}$, where $1 \leq j \leq n$). Moreover, the initial derivative information across boundary curves determines additional point coordinates in the vicinity of specified boundaries (e.g., $\mathbf{X}_{2,j}$ and $\mathbf{X}_{m-1,j}$) that are adjacent to two boundary curves at $u = 0$ and $u = 1$. Arbitrary boundary conditions can be easily enforced without any difficulty through the use of finite-difference method. Note that, in spite of certain combinations of constraint imposition shown in our experiments, in general this type of elliptic PDEs allows the boundary conditions to be explicitly formulated in arbitrary form. This permits designers to choose (various) constraints based on diverse design tasks.

3.3. Physics-based Modeling

An elastic deformable model is characterized by its positions $\mathbf{X}(u, v, t)$, velocities $\dot{\mathbf{X}}(u, v, t)$ (which stands for $\frac{\partial \mathbf{X}(u,v,t)}{\partial t}$), and accelerations $\ddot{\mathbf{X}}(u, v, t)$ (i.e., $\frac{\partial^2 \mathbf{X}(u,v,t)}{\partial t^2}$) along with material properties such as mass, damping, and stiffness distributions. These quantities are defined as functions $\mu(u, v), \gamma(u, v)$, and $\rho(u, v)$, respectively, which oftentimes can be considered to be constant at certain time. However, these material distributions are allowed to be modified by users interactively and directly over the surface model in real-time. In general, a continuous dynamic surface can be discretized into a collection of mass-points connected by a network of springs across nearest neighbors (and/or along both diagonals). Other kinds of springs (e.g., rotational springs) can be incorporated into the discretized surface if certain types of dynamic behavior is more desirable. We use a mass-spring model because of its simplicity and the critical need of real-time surface sculpting.

Applying Lagrangian mechanics, we obtain a set of second-order partial differential equations that govern the physical behavior of the underlying physics-based model:

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{D}\dot{\mathbf{X}} + \mathbf{KX} = \mathbf{f}, \tag{5}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{D}$ is a damping matrix, $\mathbf{K}$ is a stiffness matrix, and the force at every mass-point in the mesh is the sum of all possible external forces: $\mathbf{f} = \sum \mathbf{f}_{ext}$. The internal forces are generated by springs, where each spring has force $\mathbf{f}_{int} = k(\mathbf{l} - \mathbf{l}_0)$ according to Hook's law. The rest length of each spring is assigned during the PDE initialization stage. Our system allows users to modify the rest length interactively.

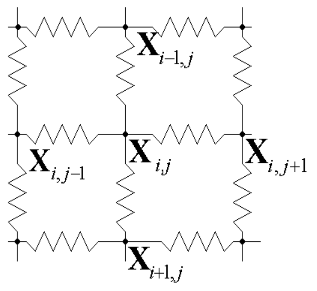

Figure 2. The mass-spring network in the vicinity of a surface point $\mathbf{X}_{i,j}$.

We associate the Lagrangian mechanics with the discretized PDE (refer to (4)) for the unified framework by attaching mass points to the geometric grid and adding springs between immediate neighbors on the rectangular PDE mesh, as shown in Fig. 2, then we obtain a dynamic version of PDE surfaces:

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{D}\dot{\mathbf{X}} + (\mathbf{K} + \mathbf{A})\mathbf{X} = \mathbf{b} + \mathbf{f}, \qquad (6)$$

where both the velocity and the acceleration of $\mathbf{X}$ can be discretized along time axis analogously:

$$\ddot{\mathbf{X}} \approx (\mathbf{X}^{t+\Delta t} - 2\mathbf{X}^{t} + \mathbf{X}^{t-\Delta t})/\Delta t^2,$$

$$\dot{\mathbf{X}} \approx (\mathbf{X}^{t+\Delta t} - \mathbf{X}^{t-\Delta t})/2\Delta t.$$

At the equilibrium, if stiffness distributions as well as the external force $\mathbf{f}$ are zero, (6) reduces to (4) with additional physical properties. By allowing a PDE surface to dynamically deform, users will have a natural feeling when interactively manipulating PDE surfaces, which is lacking without Lagrangian equations of motion. Furthermore, material properties can be introduced to govern the behavior of the underlying PDE surface. This *hybrid* formulation permits users to obtain a surface that satisfies both geometric criteria and functional requirements at the same time.

We use iteration-based techniques to solve (4) and (6). Certain variants of iteration techniques exist for solving the above linear equations [16]. We solve them using Gauss-Seidel iteration, which starts from an initial guess (approximate values) of the discretized surface points, then recursively calculates the data points in a pre-defined order. After a finite number of iterations, the value obtained through the recursive approach are considered to be extremely close to the accurate solution. To further speed up the convergence rate of Gauss-Seidel iteration, we take into account the error factor that is characterized by the difference between the approximation and the real value. This leads to the method of Successive Over-Relaxation iteration, or SOR iteration. With SOR iteration, an *Over-Relaxation factor* σ $(1 < \sigma \leq 2)$ is introduced in order to achieve a better approximation at each step. When $\sigma = 1$, it reduces to Gauss-Seidel iteration. The different choice of σ leads to different convergence speed, and the optimal value of σ is allowed to vary depending on different problems. Our system affords the interactive modification of σ, which is suitable for the solution of different kinds of linear equations. When using iterative approaches to solve the approximate linear equations of PDEs, the initial guess plays a significant role that affects the convergence speed. Hence extra cares need to be taken to ensure fewer calculations and better time performance. Furthermore, we take advantage of the multi-grid like subdivision method to speed up the numerical integration. The surface is first solved on the coarsely sampled points, and then it is refined into a finer grid whose initial values are computed either through the simple linear interpolation or more complicated subdivision schemes [10]. The convergence rate of our technique can be greatly increased. In addition, this method allows the user to control the error bound of the approximated solution.

4. Interactive Editing Toolkits

This section details various interactive techniques for PDE surface editing.

4.1. Surface Initialization

Our system supports three topological types of PDE surfaces. At the start of the initialization phase, the user must specify the surface type, i.e., whether the surface is open or closed along u and v directions. We provide users two different ways to set up the initial boundary conditions of the PDE. First, users can interactively input some control points by clicking/dragging the mouse and the system will calculate cubic B-spline curves as the boundary curves, boundary derivative curves, and other special curves that the PDE surface must interpolate. Alternatively, users are allowed to define the boundary conditions using certain implicit functions. The point coordinates are sampled along on the implicit curves, and are saved into a data file. The system then can access data files and initialize the PDE surface based on implicit function curves. After the boundary conditions are determined, the PDE surface can be derived from the solution of linear equations subject to initial conditions. If the boundary curves are B-splines, we can modify their shape by changing B-spline control points. Subsequently, the entire PDE surface will be re-computed and modified with the new boundary conditions. If the boundary curves are obtained through certain implicit functions, we can change them in the same way as adding additional conditions discussed in the following sections.

4.2. Generalized Boundary Constraints

The solution of (1) is influenced by boundary conditions. In general, there are several types of boundary conditions according to the information they contain. In this paper, we consider three kinds of boundary conditions: (1) Hermite-like conditions; (2) Coons-like conditions; and (3) Gordon-like conditions in analogy with their corresponding free-form surface formulation.

Hermite-like conditions include positions and the first or even higher derivatives of boundary curves. For the fourth-order PDE such as (1), the boundary conditions may be Hermite-like (i.e., two boundary curves at $u = 0$ and $u = 1$, and their corresponding first-order derivatives). The two boundary curves define the edges of the surface and the two derivative curves determine the gradient across the boundaries, which outline the surface shape. Fig. 4 illustrates an example that connects two PDE surfaces of this type with control function $a(u, v) = 1.5$.

For any four-sided surface patch, there are four boundary curves in general. In this case, the boundary curves are those at $u = 0$, $u = 1$, $v = 0$, and $v = 1$, respectively. This kind of conditions, in analogy with Coons patch, is considered as Coons-like boundary conditions. Using such conditions, we can easily obtain surfaces that are open along both u-direction and v-direction, or closed along v and open along u. Note that, for surfaces that are closed only along v, it is equivalent to consider that two boundary curves at $v = 0$ and $v = 1$ are the same. Fig. 5 shows an example of this type, in which we set $a(u, v) = 2.0$.

Oftentimes, it is far from enough to define complicated geometry if only four boundary curves are provided, especially when we seek the solution for PDE surfaces that are closed along both directions of u and v (e.g., tori). In this scenario, we define a curve network that the PDE surface must interpolate. This kind of boundary constraints is a direct generalization of Gordon surfaces [11]. Hence, the Gordon-like boundary conditions consist of a family of isoparametric curves $\mathbf{X}(u_i, v) = \mathbf{f}_i(v)$ and $\mathbf{X}(u, v_j) = \mathbf{g}_j(u)$, where $0 \leq u_i \leq 1$ and $0 \leq v_j \leq 1$. We show an example of this type of geometric construction (refer to Fig. 6). In Fig. 6, we specify the boundary curves at $u = 0$, $v = 0$ (which is the same as $v = 1$), $v = 0.25$, $v = 0.5$, and $v = 0.75$. The control function is $a(u, v) = 4.3$.

4.3. PDE Surface Subdivision

The large number of sample points of a PDE surface result in the slow convergence of iteration techniques. We develop a multi-grid approximation based on popular subdivision schemes to improve the computation performance. At first, we can start with a small number of sample points on the coarsest scale of a PDE surface, the coarse solution of the PDE surface can be easily obtained after a small number of iterations. Second, users can refine the coarse mesh through subdivision and use the new subdivided mesh as an initial guess for successive iterations. The finer grid is then computed iteratively to achieve a more accurate and smoother solution of the PDE surface. During the multi-grid process, the up-sampling of all boundary curves is achieved through the use of four-point interpolatory subdivision scheme [10] in order to guarantee the smoothness requirement of the refined curves. Given control points $\{\mathbf{p}_i^0\}_{i=-2}^{n+2}$, the points at level $k + 1$ of the subdivision are defined by

$$\begin{cases} \mathbf{p}_{2i}^{k+1} = \mathbf{p}_i^k \\ \mathbf{p}_{2i+1}^{k+1} = (\frac{1}{2} + w)(\mathbf{p}_i^k + \mathbf{p}_{i+1}^k) - w(\mathbf{p}_{i-1}^k + \mathbf{p}_{i+2}^k). \end{cases}$$

According to [10], the curve is tightened toward the control polygon as $w \to 0$, and for any $0 < w < (\sqrt{5} - 1)/8$, the interpolated curve is a C^1 curve. Because w influences the smoothness of boundary curves, the system allows the user to change the value of w in order to obtain satisfactory results.

4.4. Modifying Control Function

The function $a(u, v)$ can also influence surface shape. It controls the relative smoothness and the level of variable dependence between u and v directions. For a large $a_{i,j}$ at $\mathbf{X}_{i,j}$, changes in the u direction occur within a relatively short length scale, i.e., it is $1/a_{i,j}$ times the length scale in the v direction in which similar changes can take place. Consequently, the user can control how boundary conditions influence the interior of a surface by modifying the length scale (i.e., $a_{i,j}$) at arbitrary point on the PDE surface. In general, the control function $a(u, v)$ can be interactively "painted" over $\mathbf{X}_{i,j}$ (Fig. 7).

4.5. Joining Multiple Surfaces

A single PDE surface may not satisfy complicated design requirements, because real-world objects exhibit both complex topological structure and irregular geometric shape. We shall piece multiple PDE surfaces together for this purpose. In our system, users can join $n - 1$ PDE surfaces sequentially by specifying $2n$ boundary conditions (where $n \geq 3$). Note that, $2n$ conditions are necessary because two neighboring PDE patches share one common boundary. To satisfy C^1 continuity, the tangent vectors across the shared boundary must be the same. Note that, because our coefficient function $a(u, v)$ in (1) may vary throughout the u-v domain, we can consider the technique of joining multiple surfaces to be equivalent to generating one "larger" PDE surface with different local control. Fig. 4 shows such an example.

4.6. Global and Local Deformation

By changing the boundary curves, users can modify the entire shape of a PDE surface. However, when the global appearance of the PDE surface is satisfactory, any subsequent sculpting via boundary conditions may destroy certain already-existing nice features of the underlying surface. Hence, making small-scale changes on a localized region is more desirable. We enforce additional constraints to achieve this goal. Note that, the original finite-difference formulation consists of $m \times n$ equations and $m \times n$ unknowns, i.e., the coefficient matrix is a square matrix. The introduction of additional conditions forces the system to incorporate a set of new equations into the original system. Consequently, (4) becomes

$$\mathbf{A}_c \mathbf{X} = \mathbf{b}_c, \tag{7}$$

where $\mathbf{A}_c$ has $m \times n + k$ rows and $m \times n$ columns with $k > 0$. We now have more equations than the number of unknowns. There are two ways to solve such a system. One way is to treat the constraints as hard constraints, i.e., the additional equations must be satisfied. In this case, we need to explicitly formulate constraints and enforce these additional constraints within the original equations. This method works well if the additional constraints are of linear form (e.g., fixing a subset of certain unknowns or three points must be co-linear). Alternatively, one can consider additional conditions as soft constraints and solve the above equations in the least-square manner [16]. The least-square approximation is a solution of the following equations:

$$\mathbf{A}_c{}^\top \mathbf{A}_c \mathbf{X} = \mathbf{A}_c{}^\top \mathbf{b}_c.$$

Now the composite matrix becomes a square matrix, and the equations can be solved using the aforementioned techniques. Because the additional constraints discussed in this paper are all expressed as linear equations, we use the hard-constraint approach to solve the linear system. Note that, other more robust algorithms such as singular value decomposition are amenable to our PDE sculpting as well.

4.6.1 Point Editing

Our system permits users to interactively sculpt PDE surfaces by enforcing additional constraints on a set of selected points as well as their normal and curvature:

- One desirable way to manipulate a surface directly is to specify certain location in 3-space that a PDE surface must pass through. We can achieve this goal by selecting a point on the surface grid (e.g., $\mathbf{X}_{i,j}$), then dragging it to the desired position where the surface must interpolate. Moreover, users are allowed to edit a set of points, and the new and modified surface interpolates all the selected points.

- We can also manipulate the surface normal on any point to achieve local editing capability in the vicinity of the data point. The normal on a continuous surface can be approximated using the neighboring points:

$$\mathbf{n}_{i,j} = \frac{\mathbf{X}_{i+1,j} - \mathbf{X}_{i-1,j}}{2\Delta u} \times \frac{\mathbf{X}_{i,j+1} - \mathbf{X}_{i,j-1}}{2\Delta v}.$$

  When users modify the normal, our system will compute four neighboring points according to the new normal direction. In our implementation, we simply enforce four new equations within (4). The modified surface with the rotated normal at the selected point can be obtained.

- Users can also modify the curvature at arbitrary point. We consider the surface curvature at any point along u-direction and v-direction, respectively.

$$\kappa_u = \frac{\|\frac{\partial \mathbf{X}}{\partial u} \times \frac{\partial^2 \mathbf{X}}{\partial u^2}\|}{\|\frac{\partial \mathbf{X}}{\partial u}\|^3}, \kappa_v = \frac{\|\frac{\partial \mathbf{X}}{\partial v} \times \frac{\partial^2 \mathbf{X}}{\partial v^2}\|}{\|\frac{\partial \mathbf{X}}{\partial v}\|^3}.$$

  This implies that changing curvature will modify the neighboring points. If we solve the above equations directly, we need to deal with non-linear equations. To avoid this, we approximate the solution as follows. Any curvature modification reflects the distance between the two neighboring points, so we interactively edit the curvature information by attempting to move the neighboring points (e.g. $\mathbf{X}_{i-1,j}$ and $\mathbf{X}_{i+1,j}$ for κ_u at $\mathbf{X}_{i,j}$). In general, increasing the distance will reduce the curvature magnitude, while decreasing the distance will have an opposite effect on curvature value. After we compute the new position of relevant neighbors corresponding to the curvature manipulation, we can incorporate these known values of data points into the system and re-solve the equations to derive the new surface that satisfies the curvature constraints.

4.6.2 Curve Constraints

Although point-based conditions provide designers useful manipulation tools, point editing is less appropriate when users are faced with complicate design requirements. We develop editing tools that afford the intuitive specification of curve-based constraints. First, users can select a source curve on the PDE surface by picking points on the u-v domain. The curve is allowed to be of arbitrary form because the selected points may have arbitrary values of u and v, giving users more freedom for the effective editing. Second, users may specify a cubic B-spline curve as the destination curve which will then be mapped to the selected surface curve. The B-spline curve shares the same number of point samples as that of the source curve. We use B-splines because of many of its nice properties. Third, our system will attach the source curve to the destination curve, i.e., the two curves agree on the same shape. The B-spline destination curve adds a number of new linear equations into (4), and the PDE surface will be modified accordingly. Users can freely modify or even re-define a destination curve which leads to different PDE geometry. In principle, boundary conditions can be special variants of curve-based constraints. Fig. 8 illustrates an example of non-isoparametric curve constraints.

4.6.3 Region Manipulation

Certain surface models exhibit special features in specific regions, hence it is more desirable to develop region-based editing tools toward the ultimate goal of feature-based design. Analogous to the aforementioned curve tool, our system can map a user-specified B-spline destination patch onto a region of interest over the PDE surface. First, users select an area over the PDE surface. Second, users can define a B-spline patch which are sampled to have the same number of grid points as those in the source region. Third, our system attaches the specified area to the B-spline patch. Users can interactively deform the B-spline patch or create a new destination patch that imposes area constraints on the PDE geometry (see Fig. 9). Because the surface mapping algorithm depends on the structure of sample grid, we only consider source regions with rectangular grid in the interest of simplicity.

4.6.4 Area Trimming

Conventional PDE surfaces only support global manipulation, i.e., any local modification results in a new surface undergone the global deformation. This deficiency severely restrains users' freedom of arbitrary surface manipulation at any localized region(s). To overcome this difficulty, we develop a new technique that allows designers to freeze any specified area of a surface which they do not want to change. This can be achieved in our system by selecting a region in u-v domain, then any changes outside this region will not affect any data points inside. In addition, our system offers users functionalities to trim a PDE surface. After the boundary curve of a selected region is identified, users can remove material from the PDE surface either inside or outside the specified boundary. Multiple boundary curves are also available in our system, allowing the trimming on multiple regions simultaneously. The trimming operation on PDE surfaces can greatly improve the PDE surface's utility, making it possible to obtain a PDE surface with complex boundaries and of arbitrary topological type (see Fig. 10 for a trimming example).

4.7. Sculpting Dynamic PDE Surfaces

Because the run time of standard numerical solvers depends on the number of sampling points on a PDE surface, users have to patiently wait for the final stable surface as the large number of equations are solved within the system. When the number of sample points is extremely large, the computation is at the order of seconds/minutes. This significantly limits the interactivity of surface modeling and manipulation since no visual feedback between the initial and final states are provided. To ameliorate it, we consider the integrated mass-spring model of PDE surfaces whose dynamic behavior is governed by (6). The external force **f** can be computed based on various additional constraints. We then divide the time domain into small time steps and approximate both velocities and accelerations of data points through successive time intervals. We can dynamically manipulate the PDE surface with forces in real-time by solving

$$\begin{aligned}&(2\mathbf{M} + \Delta t\mathbf{D} + 2\Delta t^2\mathbf{K} + 2\Delta t^2\mathbf{A})\mathbf{X}^{t+\Delta t} = \\ &2\Delta t^2(\mathbf{b} + \mathbf{f}) + 4\mathbf{M}\mathbf{X}^t - (2\mathbf{M} - \Delta t\mathbf{D})\mathbf{X}^{t-\Delta t}.\end{aligned} \quad (8)$$

Additional constraints that control the behavior of the PDE surface can result from the editing of material properties such as mass/damping quantities and stiffness distributions. When additional constraints are incorporated into our mass-spring model, the surface points gradually evolve along consecutive time steps, hence the number of iterations to solve (8) is very small (less than 10). This results in real-time performance.

5. Implementation and Results

This section outlines the functional components of our system and presents our experimental results.

5.1. PDE Modeling Environment

We have developed a prototype software environment that permits users to interactively manipulate PDE surfaces with various constraints either locally or globally. The system is written in Visual C++ and runs on Windows95/98/NT. Fig. 3 illustrates the architecture of our prototype modeling system. In particular, our system provides the following functionalities:

Boundary Conditions. Users can interactively input and edit several types of boundary conditions, and obtain PDE surfaces satisfying these constraints. Boundary conditions can be modified freely as curve-based constraints. Moreover, the system offers a multi-grid like subdivision scheme to improve the solution.

Dynamic Models. Our system supports novel physics-based PDE surfaces including: (1) finite-difference discretization using mass-spring models; (2) multi-grid like subdivision for model refinement, and (3) finite element approximation using B-splines. Material properties and dynamic behavior greatly enhance the interactive manipulation of conventional PDE surfaces.

Sculpting Tools. Users can use various manipulation routines including: (1) patching several PDE surfaces smoothly; (2) editing points and their normal and curvature at arbitrary locations; (3) modifying boundary conditions and the control function (i.e., $a(u, v)$); (4) specifying and enforcing a set of curve constraints; (5) directly manipulating a curve network; (6) deforming a set of user-specified regions to the desired shape; (7) freezing any local region(s); (8) applying local operations only on a user-selected area; (9) trimming the specified part; and (10) modifying material properties such as mass, damping, and stiffness distributions locally.

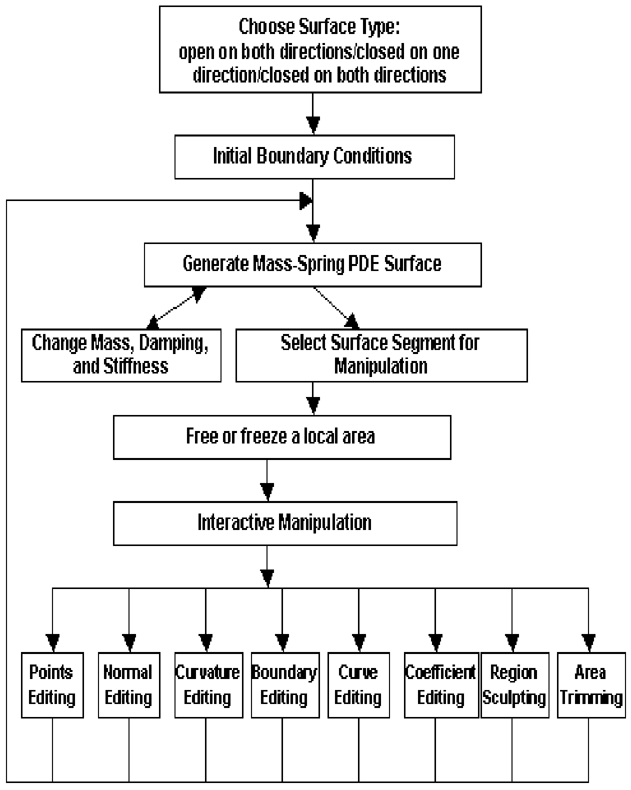
Choose Surface Type:
open on both directions/closed on one direction/closed on both directions
Initial Boundary Conditions
Generate Mass-Spring PDE Surface
Change Mass, Damping, and Stiffness
Select Surface Segment for Manipulation
Free or freeze a local area
Interactive Manipulation
Points Editing
Normal Editing
Curvature Editing
Boundary Editing
Curve Editing
Coefficient Editing
Region Sculpting
Area Trimming

Figure 3. System architecture and functionalities.

5.2. Results

We use two iterative techniques (Gauss-Seidel and SOR) to solve the PDE surface subject to various constraints. Table 1 details our experiments and their performance. Table 2 compares the CPU time of Gauss-Seidel iteration and SOR iteration with the multi-grid subdivision. We start solving the surface on 15×15 sample grid. The total CPU time to obtain a solution through the iterative approach over selected grid is the sum of the number from the coarsest level to the current level along the same column. The choice of σ will also influence the time performance in our numerical solver. We also document the CPU time of two iteration algorithms used to solve the PDE surface with different boundary conditions in Table 3.

Grid	G-S	SOR	Curve	Surface
15×15	1438	756	497	102
30×30	1751	838	1681	457
60×60	4000	1583	3504	2000

Table 1. Number of iterations for various manipulation techniques with different sampling grid. The threshold (0.001) is the sum of all distance between the corresponding points in successive steps. G-S is Gauss-Seidel iteration, SOR is SOR iteration, σ=1.25. Curve denotes curve editing with 20 sample points, while Surface stands for the regional manipulation of 10×10 sample points attached to a B-spline patch.

Grids	G-S	SOR(σ_1)	SOR(σ_2)	SOR(σ_3)
15×15	1000	109	437	593
30×30	266	79	171	282
60×60	1100	156	344	500
120×120	2954	578	750	1110

Table 2. CPU-time (ms) of different iteration techniques with the multi-grid like subdivision, where σ_1=1.05, σ_2=1.15, and σ_3=1.25.

Iteration	Fig. 4	Fig. 5	Fig. 6
G-S	8850	3365	5869
SOR	4865	2023	3786

Table 3. CPU-time (ms) of different iteration methods for solving various PDE surface examples with sample grid 30×30. σ=1.25 for the SOR method.

Besides traditional boundary conditions of the PDE techniques, our system allows users to specify and enforce a large variety of additional constraints on a set of points, cross-sectional curves, and surface regions. These constraints provide more freedom to designers, making the design process of PDE surfaces more cost-effective. We develop our prototype system using finite-difference techniques. The advantages of these approximation techniques are that they are simple, easy to implement, and suitable for the incorporation of complicated, flexible constraints. On the other hand, the time and space complexity are increased with higher resolution as well as increased accuracy. Our multi-grid like subdivision method for various levels of refinement achieves an anticipated result in our experiments.

6. Conclusion

We have presented a set of interactive algorithms that support both global and local deformation of PDE surfaces subject to generalized, flexible constraints. We have proposed a unified methodology that marries PDE surfaces with physics-based techniques. Physics-based modeling permits the PDE surfaces to be governed by physical laws and to be equipped with dynamic behavior, making the PDE surfaces more realistic and more interactive than the prior kinematic PDE surfaces. Our prototype software provides users a wide range of powerful toolkits including point-based manipulation such as position modification, normal editing, and curvature control; cross-sectional design such as boundary control and the manipulation of non-isoparametric curves; as well as region deformation such as sculpting and trimming. These enhancements permit users to model and edit PDE surfaces intuitively with ease. Our experiments have shown that generalized, flexible constraints offer users more freedom and a more natural interface to manipulate the PDE surface satisfying a set of design criteria and functional requirements. Our unified approach and novel PDE techniques greatly expand the geometric coverage and enhance the topological flexibility of conventional PDE surfaces, improving the utility of PDE surfaces for modeling and design applications, as well as helping the realization of the full potential of PDE technology in visual computing fields.

Acknowledgments

This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, and a research grant from Ford Motor Company.

References

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

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

[3] M. I. G. Bloor and M. J. Wilson. Using partial differential equations to generate free-form surfaces. *Computer Aided Design*, 22(4):202–212, 1990.

[4] M. I. G. Bloor and M. J. Wilson. Functionality in solids obtained from partial differential equations. *Computing Suppl. 8*, pages 21–42, 1993.

[5] M. I. G. Bloor and M. J. Wilson. Spectral approximations to PDE surfaces. *Computer Aided Design*, 28(2):145–152, 1996.

[6] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics*, 25(4):165–170, 1991.

[7] F. Dachille, H. Qin, A. Kaufman, and J. El-Sana. Haptic sculpting of dynamic surfaces. In *Computer Graphics (1999 SIGGRAPH Symposium on Interactive 3D Graphicsi)*, pages 103–110, Atlanta, Georgia, 1999.

[8] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[9] H. Du and H. Qin. Direct manipulation and interactive sculpting of PDE surfaces. In *Proceedings of EuroGraphics 2000*, Interlaken, Switzerland, 2000. To appear.

[10] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transaction on Graphics*, 9(2):160–169, 1990.

[11] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide*. Academic Press, 1996.

[12] T. W. Lowe, M. I. G. Bloor, and M. J. Wilson. Functionality in blend design. *Advanced in Design Automation, Vol 1: Computer Aided and Computational Design ASME, New York*, pages 43–50, 1990.

[13] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, 1991.

[14] L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. *Computer Aided Design*, 19(9):485–498, 1987.

[15] H. Qin and D. Terzopoulos. D-NURBS: A physics-based framework for geometric design. *IEEE Transaction on Visualization and Computer Graphics*, 2(1):85–96, 1996.

[16] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[17] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[18] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transaction on Graphics*, 13(2):103–136, 1994.

[19] H. Ugail, M. I. G. Bloor, and M. J. Wilson. Techniques for interactive design using the PDE method. *ACM Transaction on Graphics*, 18(2):195–212, 1999.

[20] X. Ye, T. R. Jackson, and N. M. Patrikalakis. Geometric design of functional surfaces. *Computer Aided Design*, 28(9):741–752, 1996.

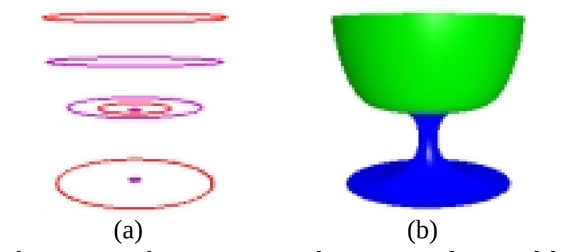
Figure 4. The connected PDE surface with Hermite-like boundary conditions: (a) boundary conditions, where boundary curves are in red and derivative curves in pink; (b) the surface subject to (a).

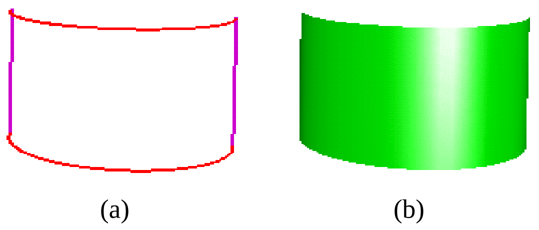
Figure 5. The open PDE surface with Coons-like boundary conditions: (a) boundary curves; (b) the corresponding surface.

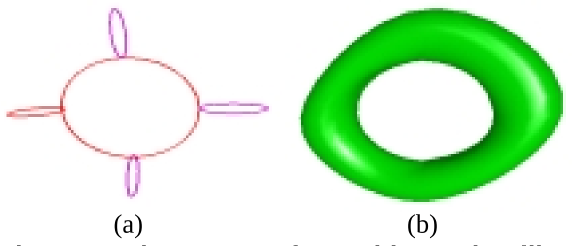
Figure 6. The PDE surface with Gordon-like boundary conditions: (a) the boundary curve network;(b) the corresponding surface.

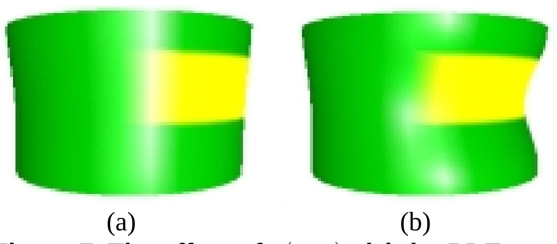
Figure 7. The effect of $a(u,v)$: (a) the PDE surface with a constant function $a(u,v) = 1.0$; (b) the surface after changing the value of $a(u,v)$ on the yellow region to 6.0.

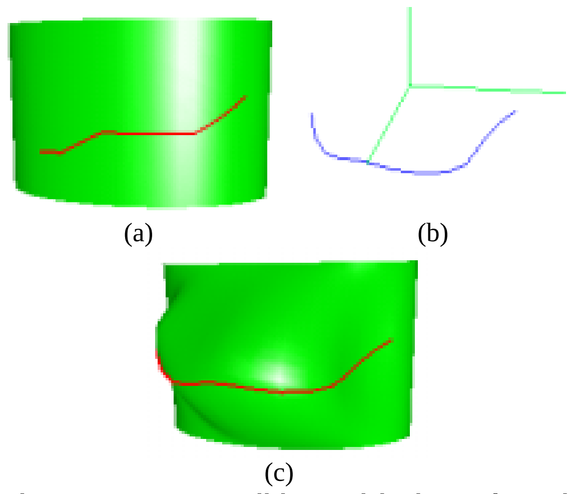
Figure 8. Curve editing: (a) the selected source curve shown in red; (b) the destination curve shown in blue; (c) the deformed PDE surface after curve attachment.

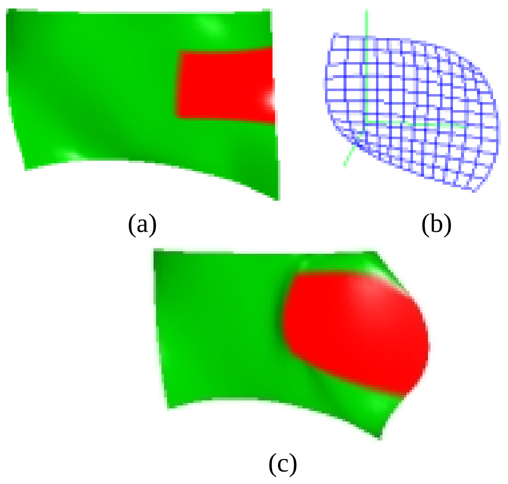
Figure 9. Region sculpting: (a) the selected source region shown in red; (b) the B-spline destination patch; (c) the deformed surface after area attachment.

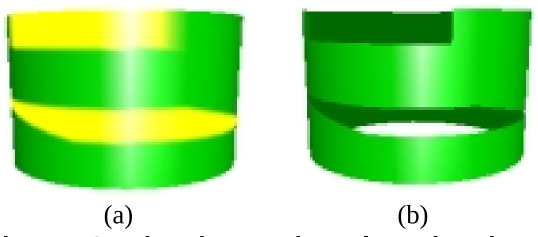
Figure 10. Trimming on the selected regions: (a) regions of interest in yellow; (b) the surface after the removal of the selected regions.