EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19 (2000), Number 3

Direct Manipulation and Interactive Sculpting of PDE Surfaces

Haixia Du Hong Qin[†]

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400

Abstract

This paper presents an integrated approach and a unified algorithm that combine the benefits of PDE surfaces and powerful physics-based modeling techniques within one single modeling framework, in order to realize the full potential of PDE surfaces. We have developed a novel system that allows direct manipulation and interactive sculpting of PDE surfaces at arbitrary location, hence supporting various interactive techniques beyond the conventional boundary control. Our prototype software affords users to interactively modify point, normal, curvature, and arbitrary region of PDE surfaces in a predictable way. We employ several simple, yet effective numerical techniques including the finite-difference discretization of the PDE surface, the multigrid-like subdivision on the PDE surface, the mass-spring approximation of the elastic PDE surface, etc. to achieve real-time performance. In addition, our dynamic PDE surfaces can also be approximated using standard bivariate B-spline finite elements, which can subsequently be sculpted and deformed directly in real-time subject to intrinsic PDE constraints. Our experiments demonstrate many attractive advantages of our dynamic PDE formulation such as intuitive control, real-time feedback, and usability to the general public.

1. Introduction and Motivation

Surface modeling and design techniques are vital to many visual computing applications including interactive graphics, interface technology, CAD/CAM, animation and entertainment, and virtual environments. Conventional modeling schemes such as Non-Uniform Rational B-Splines (NURBS) make use of simple polynomial functions in association with a multitude of control points (and weights or knots). Polynomial-based splines are very powerful and extremely popular, mainly because they are compact, easy to understand, well-behaved, and satisfying many mathematical properties such as smoothness requirements and fairness criteria. Despite the rapid advances in the theoretical foundations and mathematical properties of free-form splines during the past several decades, better and more efficient interactive techniques for popular splines have been evolved rather slowly. In essence, free-form spline modeling is frequently associated with tedious and indirect shape manipulation through time-consuming operations on a large number of (oftentimes irregular) control vertices. These conventional techniques can be difficult, less natural, and counter-intuitive, because strong mathematical sophistication is necessary for users. Moreover, only geometric attributes are characterized within mathematical splines. Recently, PDE (Partial Differential Equation) surfaces had emerged as a powerful modeling technique and started to gain popularity and strength for surface representation 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:

- PDE surfaces are generally controlled by physical laws, so they are natural and much closer to the real world. They are potentially ideal candidates for both design and analysis purposes.
- The formulation of differential equations is well-

[†] {dhaixia, qin}@cs.sunysb.edu

conditioned and technically sound. Smooth surfaces with high-order continuity requirements are readily defined through PDEs.
- Smooth surfaces that minimize certain energy functionals oftentimes are associated with differential equations, hence, optimization methods can be unified 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 process associated with PDEs, therefore, high-level 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 tasks. 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 to realize their full potential. Typical shortcomings associated with PDE surfaces include:

- The prior work on PDE surfaces mainly concentrates on elliptic PDEs and is lack of interactive techniques for shape manipulation.
- Besides boundary curve control and interactive editing of coefficients for differential equations, there is a lack of direct interactive manipulation mechanism for PDE surfaces in general.
- At present, local control is not yet to be supported. Global control is less intuitive to manipulate.

In this paper, we develop interactive techniques and implement a prototype software system to facilitate direct manipulation and interactive sculpting of PDE surfaces. Our goal is to further promote the applicability of PDE surfaces in interactive graphics and forge ahead towards the realization of the full potential for PDE surfaces. Our modeling algorithms and design framework are founded on the integrated methodology of physics-based modeling and differential equations. To present a proof-of-concept demonstration, we first discretize a continuous PDE surface using finite difference method. The initial discretization can then be interactively refined through recursive subdivision to achieve better accuracy. Later on, PDE surfaces can be approximated by B-spline finite elements. Our system offers users various interactive tools including directly manipulating normal/curvature at arbitrary location, constraining any local/global region, editing various material properties, etc. Using our system, users are able to enforce both functional requirements and geometric properties on PDE surfaces simultaneously.

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 discuss our integrated approach. Section 4 presents techniques of directly manipulating PDE surfaces. We outline our system implementation and present our experimental results in Section 5. Finally, Section 6 concludes the paper.

2. Background

We have witnessed the ever-increasing popularity of spline-based surface modeling techniques [7] in graphics applications. To date, NURBS have become an industrial standard for graphical modeling and geometric data exchange. Various NURBS-based techniques have been developed during the past twenty years such as interpolation/approximation of a set of data points and surface definition from a set of cross-sectional curves [10].

In contrast to spline surfaces, in 1989 Bloor and Wilson [1] introduced a different method that defines a smooth surface as a solution of partial differential equations (PDEs). They coined this type of surface modeling techniques as PDE surfaces. 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 certain advantages such that most of the information defining a surface comes from its boundary curves. This permits a surface to be generated and controlled through very few parameters. In addition, this PDE technique can be used to generate piecewise free-form surfaces [2]. By varying boundary conditions and control parameters in the PDE, designers can obtain various surface shapes. Furthermore, Lowe, Bloor and Wilson [11] 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 is possible to simultaneously introduce geometric constraints, aesthetic criteria, and physical and engineering restrictions into the design process. Additionally, Bloor and Wilson [3] developed an algorithm that approximates PDE surfaces using standard B-splines. Their work intends to demonstrate that PDE surfaces are virtually compatible with other matured and well established techniques based on popular splines 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 by Bloor and Wilson [4], which further expands the geometric coverage of PDE methodology. For certain simple boundary conditions, the 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 solutions for PDEs in closed form for 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 which satisfy PDEs with an additional 'corrector' term that satisfies boundary conditions.

However, the aforementioned 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 physics-based modeling methodology with PDE approach, mainly because that the dynamic behaviors of physics-based models are also controlled by differential equations (e.g., Lagrangian equations of motion). Terzopoulos and Fleischer [14] 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 in Terzopoulos and Fleischer [14]. Terzopoulos and Qin [12, 15] formulated a novel model for interactive sculpting Dynamic NURBS (D-NURBS). Ye, Jackson and Patrikalakis [16] incorporated certain functional constraints into the design process of geometric shapes. Dachille *et al.* [9] presented an approach for the direct manipulation of physics-based B-spline surfaces. In essence, because most physical phenomena can be described in the form of PDEs, it is readily feasible to integrate geometric PDE surfaces with physics-based modeling approaches to achieve the full potential of PDE surfaces.

3. Formulation and Properties

This section formulates PDE surfaces, and discusses properties of the integrated approach of PDE surfaces and physics-based modeling.

3.1. Differential Equations

Throughout this paper, we focus on solving the fourth-order elliptic PDEs [1]:

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

where u, v are two parametric coordinates, a is a control parameter, and $\mathbf{X}(u,v) = \left[\begin{array}{ccc} x(u,v) & y(u,v) & z(u,v) \end{array}\right]^{\top}$ denotes the point coordinates of the PDE surface in 3-space. Note that, our mathematical derivation and its associated numerical techniques can be readily generalized to other arbitrary PDEs. To simplify the matter, we consider that four boundary conditions comprise two curves which define a pair of the curved surface edges at the opposite side along one parametric direction (e.g., u) and a pair of their associated derivative curves defining gradient information across the two curved edges throughout this paper. Without loss of generality, we only consider PDE surfaces that are geometrically closed along the second parametric direction (e.g., v). For the convenience of the following discussion, we allow the value of u to vary between 0 and 1, and v to vary between 0 to 2π. The boundary conditions are of the following form:

$$\mathbf{X}(0,v) = \mathbf{c}_0(v), \mathbf{X}(1,v) = \mathbf{c}_1(v),$$

$$\frac{\partial \mathbf{X}}{\partial u}(0,v) = \mathbf{c}'_0(v), \frac{\partial \mathbf{X}}{\partial u}(1,v) = \mathbf{c}'_1(v). \qquad (2)$$

By interactively modifying boundary curves and their gradient curves, users are capable of manipulating the entire surface in an *indirect* manner. This gives the designer an efficient way to edit the PDE surface through a fewer number of parameters that define boundary curves.

3.2. Physics-based Modeling

A deformable model is characterized by its positions $\mathbf{X}(u,v,t)$, velocities $\dot{\mathbf{X}}(u,v,t)$, and accelerations $\ddot{\mathbf{X}}(u,v,t)$ along with material properties such as mass, damping, and stiffness distributions. Applying Lagrangian mechanics, we obtain a set of 2nd-order nonlinear differential equations that govern the physical behavior of the underlying physics-based model:

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{D}\dot{\mathbf{X}} + \mathbf{K}\mathbf{X} = \mathbf{f}, \qquad (3)$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{D}$ is a damping matrix, $\mathbf{K}$ is a stiffness matrix, and $\mathbf{f}$ represents the entire external force applied on the surface.

3.3. Numerical Techniques

Prior work on PDE surfaces mainly seeks closed-form explicit solutions (e.g., Fourier series functions) in order to exploit many attractive properties of analytic formulations for surface design. In the interest of arbitrary boundary conditions, we resort to numerical techniques that guarantee a solution of the integrated formulation, unifying both dynamic models and PDE surfaces within a single design framework.

Numerical algorithms facilitate the material modeling of anisotropic distribution and its realistic physical simulation, where there exist no accurate analytic solutions for PDE surfaces. Among many matured techniques, we employ two popular numerical approaches to demonstrate the universal applicability of our framework: (1) finite-difference discretization, and (2) finite-element method based on B-spline approximation (see Sec. 4.6 for the details).

The finite-difference method transforms a partial differential equation to a system of algebraic equations by replacing all partial derivatives in the differential equation with their discretized approximations. The system of algebraic equations can then be solved numerically either through an iterative process or a direct procedure to obtain an approximate solution to the continuous PDE. Based on the Taylor series expansion of a continuous function $f(x)$, we derive the central-difference approximation of $f'(x)$: $f'(x) = (f(x+h) - f(x-h))/2h$, as well as the approximation of $f''(x)$: $f''(x) = [f(x+h) - 2f(x) + f(x-h)]/h^2$, where h denotes the grid interval. We can easily extend the computation of univariate derivatives to all partial derivatives of bivariate surface geometry, by dividing the $[u, v]$ domain into m and n discretized points, respectively. Now, (1) can be rewritten as:

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

where $\mathbf{A}$ is the discretized differential operator in a $(m \times n) \times (m \times n)$ matrix form, and

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \cdots & \mathbf{x}_{m,n} \end{bmatrix}^\top,$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_{m\times n} \end{bmatrix}^\top.$$

Typical boundary conditions permit us to derive the point coordinates lying on the curved edges of $u = 0$ and $u = 1$ (i.e. the value of $\mathbf{x}_{1,j}$ and $\mathbf{x}_{m,j}(1 \le j \le n)$, respectively. Moreover, given the initial derivative information along two opposite curves, we can explicitly compute additional $2n$ data points in the discretization. These data points lie on two curves (i.e., $\mathbf{x}_{2,j}$ and $\mathbf{x}_{m-1,j}$) that are adjacent to two boundary curves. In analogy with the original PDE [1], we also assume that the solution surface is closed along its v direction, so the points on $v = 0$ are the same as those on $v = 2\pi$. We solve the set of linear equations whose unknowns are $\mathbf{x}_{i,j}$ using linear algebra techniques. Despite the specific constraints we enforce for the PDE surface used in this paper, 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.

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

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{D}\dot{\mathbf{X}} + (\mathbf{K} + \mathbf{A})\mathbf{X} = \mathbf{b} + \mathbf{f}, \tag{5}$$

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

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

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

Note that at the equilibrium, if stiffness distribution as well as the external force vector $\mathbf{f}$ are zero, (5) reduces to (4) with additional physical properties such as mass and damping distributions. By allowing the PDE surface to dynamically deform, users will have a natural feeling when they interactively manipulate the PDE surface, 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 affords users to obtain a surface that satisfies both geometric criteria and functional requirements at the same time.

In general, we either resort to direct approaches or make use of iteration-based techniques. Certain variants of iteration techniques exist for solving the above linear equations [13]. We solve them using Gauss-Seidel iteration, which starts from an initial guess of the discretized surface points, then recursively calculates the data points in a pre-defined order. After a finite number of steps, the values obtained through the recursive approach are considered to be extremely approaching 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 Successive Overrelaxation (SOR) iteration. When using iterative approaches to solve the approximated linear equations of the PDE, 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 [8]. The convergence rate of our technique is greatly increased. In addition, this method allows the user to control the error bound of the approximated solution.

4. Interactive Techniques

This section details various interactive techniques and explains the implementation issues.

4.1. PDE Surface Initialization

We provide users two different ways to initialize a PDE surface. First, users can interactively input control points using mouse and the system will calculate cubic B-spline curves as the boundary curves. Then using the same method, users can enter control points to obtain two extra curves, corresponding to the two boundary curves, respectively. The difference between any point on each newly defined curve and its associated point on the boundary curve will be used to determine both the magnitude and the direction of the derivative vector across the two boundary curves. Alternatively, users are allowed to define the boundary curves and their derivatives using certain implicit functions. The point coordinates are sampled along the implicit curves, and saved into a data file. The system can retrieve the data file and initialize the PDE surface based on implicit function curves. The PDE surface then can be derived based on the solution of linear equations subject to initial values explained above (refer to Fig. 1 and Fig. 2).

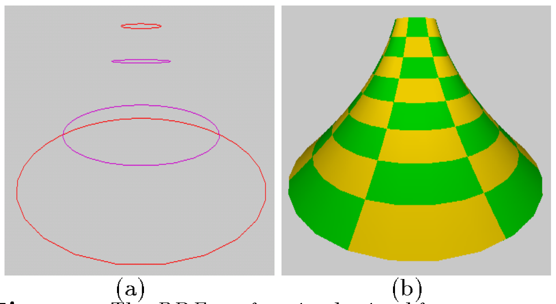
Figure 1: *The PDE surface is obtained from curves of implicit functions: (a) Boundary conditions obtained from a file; (b) The PDE surface subject to (a)*

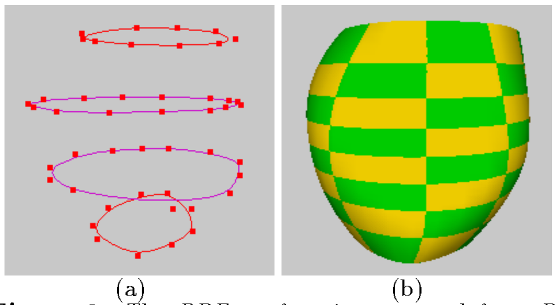
Figure 2: *The PDE surface is generated from B-spline curves: (a) B-spline boundary curves with control points; (b) The corresponding PDE surface.*

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. Modifying Coefficient a

The coefficient a can also influence the global shape of a surface. This coefficient controls the relative smoothness and the level of variable dependence between the u and v directions. For a large a, changes in the u direction occur within a relatively short length scale, i.e. it is $1/a$ times the length scale in the v direction in which similar changes can take place. As a result, users can control how boundary conditions influence the interior of a surface by modifying the length scale (i.e. a). In general, the coefficient a is chosen independently of u and v. We use $a = 2$ in this paper for our experiments.

4.3. Joining Multiple PDE Surfaces

Oftentimes a single PDE surface can 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. Fig. 3 illustrates how to connect several PDE surfaces with six boundary curves. There are seven curves after the gradient curve on one side of the shared boundary is calculated from another one on the opposite side to obtain C^1 continuity.

4.4. Enforcing Additional Conditions

By changing the boundary curves, users can modify the entire shape of a PDE surface. However, when the global appearance of a PDE surface is satisfactory, any subsequent sculpting via boundary conditions may destroy certain already-existing nice features of the underlying surface. In this scenario, 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

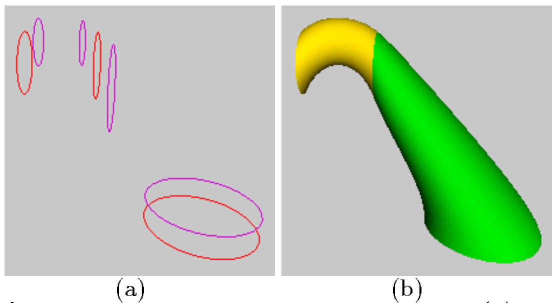

Figure 3: *Connecting two PDE surfaces: (a) The boundary conditions; (b) The connected composite surface.*

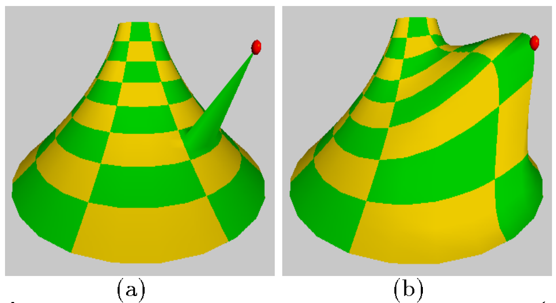

Figure 4: *Moving one point on the PDE surface: (a) The point editing; (b) The modified PDE surface.*

of additional conditions forces the system to incorporate a set of new equations into the original set. As a result, (4) becomes

$$\mathbf{A}'\mathbf{X} = \mathbf{b}', \tag{6}$$

where $\mathbf{A}'$ 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 with those additional constraints. 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 a least-square fashion [13]. The least-square approximation is a solution of the following equations:

$$\mathbf{A}'^{\top}\mathbf{A}'\mathbf{X} = \mathbf{A}'^{\top}\mathbf{b}'. \tag{7}$$

Now the composite matrix becomes a square matrix, and the equation can be solved using the aforementioned techniques in this paper (Note that, other more robust algorithms such as singular value decomposition are amenable to our PDE sculpting as well).

4.4.1. Manipulating Surface Points

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 grids (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 data points. Fig. 4 shows the modified PDE surface by changing the position of one point on the original surface.

4.4.2. Changing Surface Normal

We can also manipulate the surface normal on any point to achieve a local editing capability in the vicinity of the data point, as demonstrated in Fig. 5. The normal on a continuous surface can be approximated by 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 point 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 a rotated normal at the selected point can be obtained. An example for this kind of constraints is shown in Fig. 6.

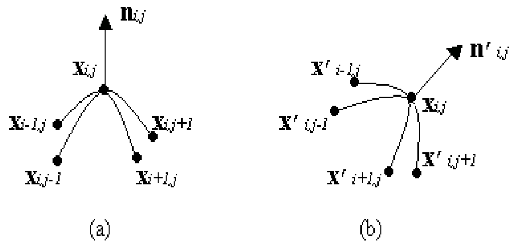

Figure 5: *Changing the point normal results in the change of neighbor points: (a) Normal $\mathbf{n}_{i,j}$ of $\mathbf{x}_{i,j}$; (b) The new normal $\mathbf{n}'_{i,j}$.*

4.4.3. Editing Surface Curvature

Users can also change the shape of a PDE surface by modifying the curvature at arbitrary point. The curve curvature is: $\kappa = \frac{\|x' \times x''\|}{\|x'\|^3}$. We consider the surface curvature at any surface point along u-direction and v-direction, respectively.

$$\kappa_u = \frac{\|\frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial^2 \mathbf{x}}{\partial u^2}\|}{\|\frac{\partial \mathbf{x}}{\partial u}\|^3}, \kappa_v = \frac{\|\frac{\partial \mathbf{x}}{\partial v} \times \frac{\partial^2 \mathbf{x}}{\partial v^2}\|}{\|\frac{\partial \mathbf{x}}{\partial v}\|^3}. \tag{8}$$

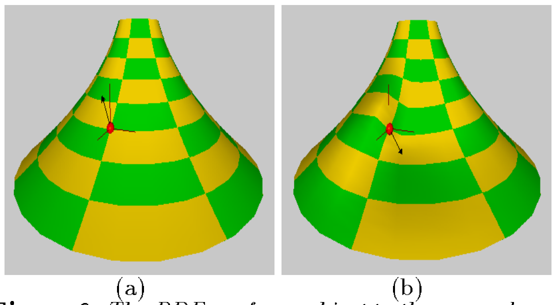

Figure 6: *The PDE surface subject to the normal constraint: (a) The black line with arrow is the red point's normal, red lines are local coordinate system; (b) The modified surface.*

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 (refer to Fig. 7). Any curvature modification reflects the

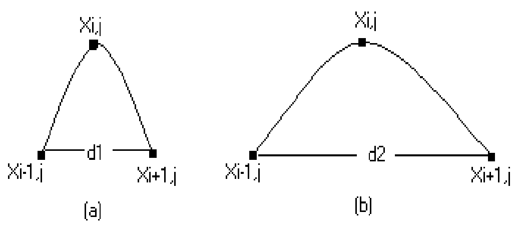

Figure 7: *Curvature modification via the change of the distance between the neighboring points: (a) High curvature due to short distance; (b) Low curvature due to long distance*

distance between the two neighboring points, so we interactively edit the curvature information by attempting to move the neighboring points $\mathbf{x}_{i-1,j}$ and $\mathbf{x}_{i+1,j}$. 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-compute the equations to derive the new surface that satisfies the curvature constraints. In Fig. 8, we modify a PDE surface with curvature constraints.

4.4.4. Sculpting Localized Regions

Standard 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

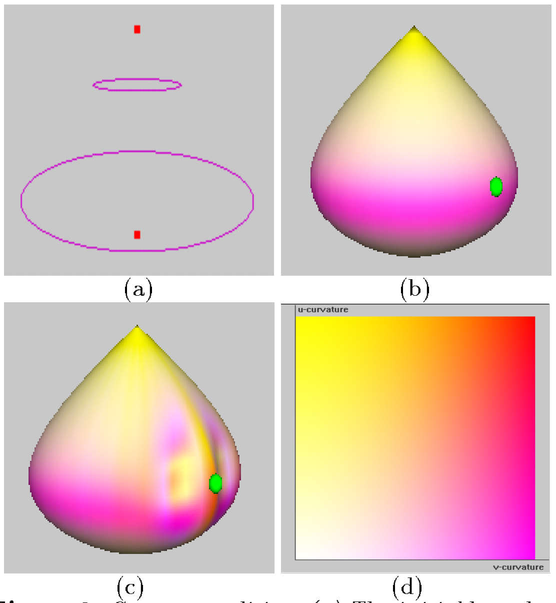

Figure 8: *Curvature editing: (a) The initial boundary conditions for a PDE surface; (b) The surface with curvature mapping; (c) The surface after changing u-curvature* $0.8 \to 4.2$, *v-curvature* $1.3 \to 4.1$ *on a point. (d) The curvature map in u-v domain.*

manipulation at any localized region(s). To overcome this difficulty, we develop a new technique that allows the designer to fix any specified area of a surface which he/she does not want to change. This can be achieved in our system by selecting a region in the parametric domain of u and v, then any changes outside this region will not affect any data points inside, as shown in Fig. 9.

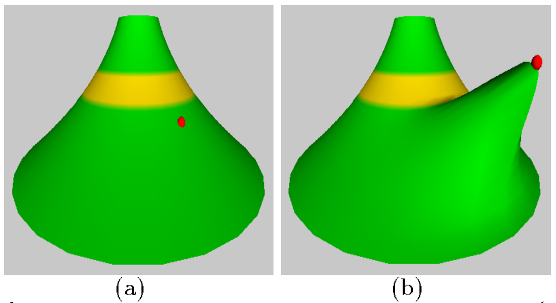

Figure 9: *Surface manipulation with fixed regions: (a) The yellow part is fixed; (b) The modified surface without disturbing the fixed part.*

4.5. Forces

Because the run time of standard numerical solvers depends on the number of sampling points on the PDE surface, users oftentimes 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 are tremendous, the computation time is rather long at the order of seconds/minutes. This significantly limits the interactivity of surface modeling and manipulation as no visual feedback between the initial and final states are provided. To ameliorate, we can implicitly compute the external force $\mathbf{f}$ based on various additional constraints as discussed above. We then divide the time domain into many small time steps and approximate both the velocity and the acceleration of data points through successive time intervals. We can dynamically manipulate the PDE surface with forces in real-time by solving

$$\begin{aligned}(2\mathbf{M} + \Delta t\mathbf{D} + 2\Delta t^2\mathbf{K} + 2\Delta t^2\mathbf{A})\mathbf{X}^{t+\Delta t} &= \\ 2\Delta t^2(\mathbf{b} + \mathbf{f}) + 4\mathbf{M}\mathbf{X}^t - (2\mathbf{M} - \Delta t\mathbf{D})\mathbf{X}^{t-\Delta t}\end{aligned} \quad (9)$$

Additional constraints that control the behavior of the PDE surface can be obtained via the editing of material properties such as mass/damping quantities and stiffness distribution. When additional constraints are incorporated into our mass-spring model, between consecutive time steps the data positions gradually evolve, hence the number of iterations to solve (9) is very small (less than 10). This results in real-time performance.

4.6. B-spline Approximation

To facilitate the data exchange capability of PDE surfaces with standard spline-based systems, we compute a B-spline approximation of the PDE surface. A B-spline surface over u, and v domain can be defined as

$$\mathbf{X}(u,v) = \sum_{i=1}^{k}\sum_{j=1}^{l} B_{i,c}(u)B_{j,d}(v)\mathbf{p}_{i,j}, \quad (10)$$

where $B_{i,c}(u)$ and $B_{j,d}(v)$ are B-spline basic functions of u and v with the order c and d, respectively, $\mathbf{p}_{i,j}(1 \le i \le k, 1 \le j \le l)$ are B-spline control points. Oftentimes the number of control points is less than the number of sample points on the PDE surface, therefore the B-spline approximation results in a family of over-constrained linear equations whose unknowns are fewer than the number of equations. For example, given the $m \times n$ sample points on the PDE surface, the approximation using $k \times l$ control points leads to

$$\mathbf{BP} = \mathbf{X}, \quad (11)$$

where there are $m \times n$ linear equations with $k \times l$ unknowns. Assuming fixed parameterization of data points in B-spline approximation, the matrix $\mathbf{B}$ is a discretization of basis functions. This over-constrained system can be solved by multiplying $\mathbf{B}^\top$ on both sides of (11). Consequently, we obtain a B-spline surface that approximates the PDE surface in a least-square sense. Fig. 10 shows a B-spline approximation for previous examples. Meanwhile, we also use B-spline finite elements to approximate the dynamic model of PDE surfaces at each time step. This allows users to interactively manipulate the B-spline solution of PDE surfaces with forces in real-time.

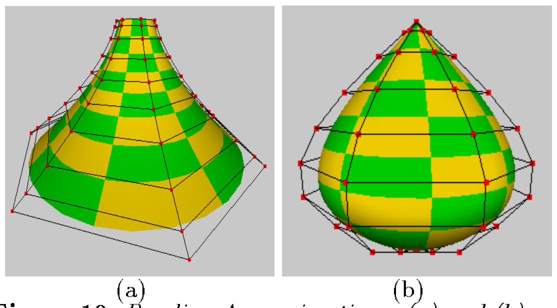

Figure 10: *B-spline Approximations: (a) and (b) are two approximations for Fig. 1 and Fig. 8, respectively. The black lines with red points are the B-spline control mesh.*

5. System Implementation

This section outlines the system architecture, its functional components and details the set of design toolkits.

5.1. Architecture and Toolkits

We have developed a prototype software system that permits users to interactively manipulate PDE surfaces either locally or globally. The system is written in Visual C++ and runs on PCs. Fig. 11 illustrates the architecture of our prototype system. Our system provides the following functionalities:

Boundary Curves. Users can interactively input and edit control points of cubic B-spline curves as boundary conditions. Commonly-used implicit functions can also be retrieved from a data file to serve as boundary curves during the initialization phase. Users can obtain a PDE surface satisfying these constraints.

Dynamic Models. Our system supports novel physics-based PDE surfaces including: (1) finite-difference discretization using mass-spring models; (2) multigrid-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 are free to use various manipulation routines including: (1) joining several PDE

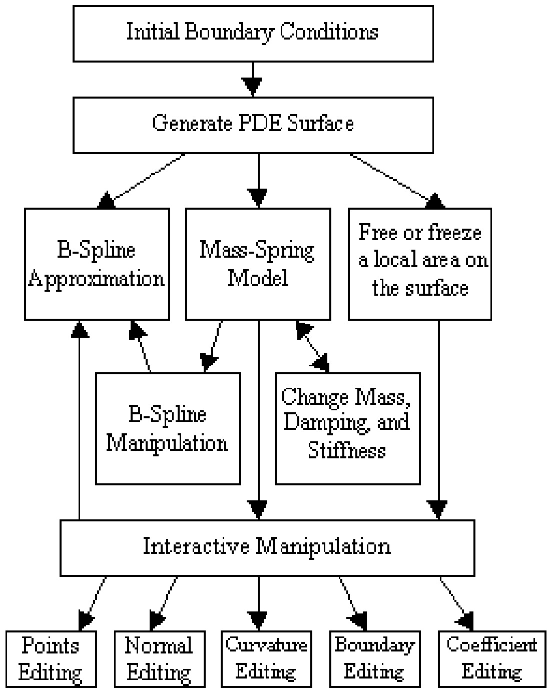

Figure 11: *System architecture*

Grids	G-S	SOR	P	N	C
15×15	1438	756	253	146	447
30×30	1751	838	1420	146	494
60×60	4000	1583	933	146	190

Table 1: *Number of iterations for various manipulation techniques with different sampling grids. The iteration threshold (0.001) is the sum of all distance between the corresponding points in successive steps. G-S stands for Gauss-Seidel iteration, SOR stands for SOR iteration. P, N, and C denote point, normal and curvature manipulations on the surface, respectively.*

Example	Grids	Mesh	μ	γ	ρ	Δt
(a)	30×30	N/A	20	70	100	0.1
(b)	30×30	8×8	20	70	100	0.1

Table 2: *The physical parameters in several dynamic surfaces. Mesh stands for grids of the B-spline control mesh, and μ, γ, and ρ represent Mass, Damping and Stiffness distribution.*

surfaces smoothly; (2) moving (a set of) arbitrary surface points to desired locations; (3) modifying surface normals at arbitrary surface points; (4) editing surface curvatures at arbitrary surface points; (5) changing boundary conditions and the coefficient (i.e., a) associated with the PDE; (6) specifying and fixing any local region(s) of interests; (7) manipulating geometric attributes only in a user-specified area; (8) modifying material properties such as mass, damping, and stiffness distributions in any user-specified region(s); (9) computing the B-spline approximation of PDE surfaces; and (10) directly deforming B-spline finite elements with "forces".

5.2. Results and Discussion

We use several numerical techniques to solve the PDE surface subject to various constraints. Table 1 details our experiments and their performance. Note that, it generally takes more iterations in a coarsely sampled grid, however, the CPU time spent on the coarser grid is far less than that on the finer grid. In Table 2, we summarize the physical parameters used in our mass-spring examples as well as the B-spline approximation for mass-spring PDE surfaces in Fig. 12.

We enforce additional constraints beyond conventional boundary conditions of PDE surfaces. These constraints provide more freedom to designers when modeling a PDE surface, making it more efficient to achieve desired results. We use both finite-difference and B-spline finite element techniques for implementation. The advantages of these approximation techniques are that they are simple, easy to implement, and suitable for additional constraints. On the other hand, the time and space complexity are increased with higher resolution as well as increasing accuracy. And the convergence speed of the iteration depends on the initial values. Our multi-grid like subdivision method for various levels of refinements achieves an anticipated result in our examples.

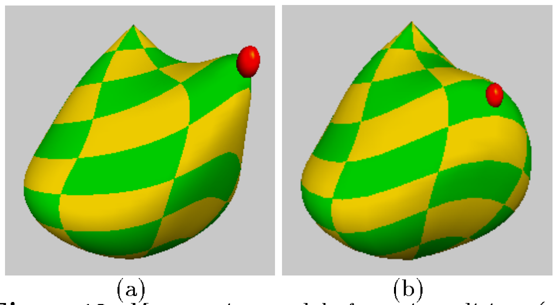

Figure 12: *Mass-spring models for point-editing: (a) The mass-spring model of the PDE surface; (b) The mass-spring model of the B-spline surface.*

6. Conclusion

We have presented a set of interactive techniques that can incorporate additional geometric constraints and material criteria to the conventional PDE method. Our prototype software provides users a wide range of powerful toolkits such as point manipulation, normal editing, curvature control, local sculpting, as well as boundary control. These value-added capabilities permit users to model and manipulate PDE surfaces intuitively. In addition, we have integrated the PDE surfaces with our physics-based modeling framework. One major advantage of the PDE method is that users can deploy fewer design constraints to achieve desired design effects. Our experiments have shown that these additional constraints offer users more freedom and a more natural interface to manipulate the PDE surface satisfying a set of design criteria and functional requirements. Furthermore, physics-based modeling permits the PDE surface to be governed by physical laws and to be equipped with material properties, making the PDE surface more realistic and more interactive than the prior kinematic PDE surface. Our system also computes the B-spline finite element approximation of the PDE surface and allows users to interactively manipulate B-splines to support the data exchange capability in commercially available geometric modeling systems. The future directions comprise: (1) the generalization to other types of PDEs; (2) the design and analysis of more effective and robust algorithms for PDE surfaces; (3) the development of more complicated curve/regional constraints and toolkits; (4) a as a bivariate function of u and v; (5) open/close PDE surfaces with/without self-intersection; (6) the integration of subdivision surfaces with PDEs for complex boundary conditions of arbitrary topology and multi-resolution analysis; etc.

Acknowledgment

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), pp. 165-171, 1989. 2, 3, 4

2. M.I.G. Bloor and M.J. Wilson. Using Partial Differential Equations to Generate Free-Form Surfaces, *Computer Aided Design*, **22**(4), pp. 202-212, 1990. 2

3. M.I.G. Bloor and M.J. Wilson. Representing PDE surfaces in Terms of B-splines, *Computer Aided Design*, **22**(6), pp. 324-331, 1990. 2

4. M.I.G. Bloor and M.J. Wilson. Functionality in Solids Obtained from Partial Differential Equations, *Computing Suppl. 8*, pp. 21-42, 1993. 2

5. M.I.G. Bloor and M.J. Wilson. Spectral Approximations to PDE surfaces, *Computer Aided Design*, **28**(2), pp. 145-152, 1996. 3

6. G. Celniker and D. Gossard. Deformable Curve and Surface Finite Elements for Free-Form Shape Design, *Computer Graphics*, **25**(4), pp. 165-170, 1991. 3

7. C. de Boor. A Practical Guide to Splines, *Springer*, 1978. 2

8. N. Dyn, D. Levin and J.A. Gregory. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control, *ACM Transaction on Graphics*, **9**(2), pp. 160-169, 1990. 4

9. F. Dachille IX, H. Qin, A. Kaufman, and J. El-Sana. Haptic Sculpting of Dynamic Surfaces, *1999 Symposium on Interactive 3D Graphics, Atlanta, Georgia*, pp. 103-110, 1999. 3

10. L. Piegl. On NURBS: A Survey, *IEEE Computer Graphics and Applications*, **11**(1), pp 55-71, 1991. 2

11. 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*, pp. 43-50, 1990. 2

12. H. Qin and D. Terzopoulos. D-NURBS: A Physics-Based Framework for Geometric Design, *IEEE Transaction on Visualization and Computer Graphics*, **2**(1), pp. 85-96, 1996. 3

13. G. Strang. Introduction to Applied Mathematics, *Wellesley-Cambridge Press, 1986.* 4, 6

14. D. Terzopoulos and K. Fleischer. Deformable Models, *The Visual Computer*, **4**(6), pp. 306-331, 1988. 3

15. D. Terzopoulos and H. Qin. Dynamic NURBS with Geometric Constraints for Interactive Sculpting, *ACM Transaction on Graphics*, **13**(2), pp 103-136, 1994. 3

16. X. Ye, T. R. Jackson, and N. M. Patrikalakis. Geometric Design of Functional Surfaces, *Computer Aided Design*, **28**(9), pp. 741-752, 1996. 3