ELSEVIER

Computer-Aided Design 36 (2004) 1101–1116

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

# A shape design system using volumetric implicit PDEs

Haixia Du*, Hong Qin

*Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY, USA*

Received in revised form 12 October 2003; accepted 9 January 2004

**Abstract**

Solid modeling based on partial differential equations (PDEs) can potentially unify both geometric constraints and functional requirements within a single design framework to model real-world objects via its explicit, direct integration with parametric geometry. In contrast, implicit functions indirectly define geometric objects as the level-set of underlying scalar fields. To maximize the modeling potential of PDE-based methodology, in this paper we tightly couple PDEs with volumetric implicit functions in order to achieve interactive, intuitive shape representation, manipulation, and deformation. In particular, the unified approach can reconstruct the PDE geometry of arbitrary topology from scattered data points or a set of sketch curves. We make use of elliptic PDEs for boundary value problems to define the volumetric implicit function. The proposed implicit PDE model has the capability to reconstruct a complete solid model from partial information and facilitates the direct manipulation of underlying volumetric datasets via sketch curves and iso-surface sculpting, deformation of arbitrary interior regions, as well as a set of CSG operations inside the working space. The prototype system that we have developed allows designers to interactively sketch the curve outlines of the object, define intensity values and gradient directions, and specify interpolatory points in the 3D working space. The governing implicit PDE treats these constraints as generalized boundary conditions to determine the unknown scalar intensity values over the entire working space. The implicit shape is reconstructed with specified intensity value accordingly and can be deformed using a set of sculpting toolkits. We use the finite-difference discretization and variational interpolating approach with the localized iterative solver for the numerical integration of our PDEs in order to accommodate the diversity of generalized boundary and additional constraints.



*Keywords:* Partial differential equation techniques; Implicit functions; Volume graphics; Shape design; Geometric constraints; Scattered data fitting

## 1. Introduction

Partial differential equation (PDE) techniques are widely used for many visual computing applications, such as nature phenomena simulation and animation [17], variational fairing [35], image inpainting [2], etc. They also provide an alternative way for geometric design [5–7,45]. Different from traditional geometric representations, the PDE methods model graphical objects as solutions of certain elliptic PDEs with boundary constraints inside the parametric domain. The parametric PDE model simplifies the geometric design process by using only boundary conditions to recover the whole interior information and offers high-order continuity as well as energy minimization properties. However, it is extremely difficult to model arbitrary shapes of general topology, because the PDEs are defined over regular parametric domain, like traditional parametric approaches.

In contrast, implicit functions use level-sets of certain scalar field functions in the physical domain directly to design, model, and interact with 3D objects, without constructing the mapping between parametric and physical spaces. They offer a fundamentally different yet convenient and natural design paradigm (in comparison with parametric representations) in visual computing fields such as graphics, animation, and geometric design. This is because of their unique properties such as arbitrary topology, collision detection, free of parametric correspondence, etc. Applications of implicit functions include shape blending, surface reconstruction from scattered data points, shape transformation, and interactive modeling [3,4,8–10,12,13,18,21,22,26,27,29,30,32,36,39,40,43,44,47].

Implicit functions offer several modeling advantages such as flexible topology, simple data structure, efficient

* Corresponding author.
*E-mail addresses:* dhaixia@cs.sunysb.edu (H. Du); qin@cs.sunysb.edu (H. Qin).

0010-4485/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2004.01.009

storage, volumetric information, unbounded geometry, etc. Nonetheless, most of implicit functions focus on surface models. Previous techniques for interactive implicit volume sculpting have certain modeling limitations. Recently, Cutler et al. [11] presented a procedural framework for specifying layered solid models and applying a series of simulation operations (serving as sculpting tools described by a script language) to complex models. Bærentzen and Christensen [1] developed an interactive volume sculpting system using level-set method. Museth et al. [28] proposed level-set-based editors for CSG operations, blending, embossing, and smoothing for implicit surfaces. However, these tools are associated with the specification of speed functions for the evolving level-set, which are non-intuitive for common users. Turk and O'Brien [41] presented interactive implicit surface sculpting via particles, but each operation requires reformatting and recalculation of the entire system, which is difficult to model large datasets. In general, the modeling potential of implicit functions has not been fully explored yet and there are still difficulties to design, reconstruct, and sculpt implicit models directly and intuitively.

To maximize the modeling capabilities of PDE techniques and implicit functions in geometric and visual computing areas, we propose a more general PDE-based modeling paradigm which integrates the PDE techniques with implicit functions into one single framework for interactive shape design and manipulation on PDE-based volumetric implicit models. We develop an implicit modeling system governed by elliptic PDEs of scalar intensity fields. In particular, our prototype system can reconstruct implicit objects and the embedding implicit 3D working space as approximated solutions of the PDEs by specifying a set of curve outlines or scattered data points of certain intensity values as *general* boundary constraints with the assistance of variational interpolating approaches. Because the curves and datasets are not required to be closed, open surfaces can be modeled within our system. Moreover, it offers a set of sculpting toolkits to manipulate implicit objects, such as interactively modifying the geometric shape, intensity value, and gradient direction of selected sketch curves, directly changing intensity values of selected regions in the working space, as well as deforming iso-contours at specified intensity values of the objects. Because the working space is governed by the PDEs, any missing information inside the space can be recovered by solving the PDEs according to the given constraints. Our system is able to recover damaged datasets using partial information, smooth the intensity distribution of volume data, and smoothly blend objects inside the working space. In general, our system allows intensity manipulations at any iso-value anywhere in the implicit working space to model implicit objects either directly or indirectly, which offers users both local and global control of the implicit PDE model.

This implicit PDE approach has modeling advantages of both parametric PDE techniques and implicit functions. First, the behavior of the implicit PDE model is governed by differential equations. Solving the PDEs results in both boundary and interior information simultaneously, which offers an alternative way to model implicit objects by using only boundary information. This property makes the PDE method extremely suitable for shape blending process. Second, many natural physical processes are characterized by differential equations in principle [19,20,37]. Hence, PDE models are natural and close to the real world. They are potentially ideal candidates for design, simulation, and analysis tasks. Furthermore, geometric objects with high-order continuity requirements can be readily defined through high-order PDEs because of their differential properties. Third, smooth objects that minimize certain energy functionals are the solutions of differential equations from the variational analysis point of view, so optimization techniques can be unified with PDE models. In addition, because the implicit PDE is formulated on a scalar intensity field and defines objects by collecting points of certain iso-values, it is capable of designing arbitrary topological shapes and recovering the full information from partial input, which reduces the burden of specifying the large quantity of constraints for complete datasets. It offers users a natural way to design objects easily with general non-isoparametric arbitrary curve outlines, reconstruct objects from scattered data points, blend shapes in the working space, and recover damaged datasets.

The remainder of the paper is structured as follows. Section 2 reviews the related work of PDE techniques and implicit models. We detail the PDE formulation and present our integrated approach for implicit PDE objects in Section 3. We introduce possible applications of our implicit PDE model by enforcing different types of boundary and additional constraints in Section 4. Section 5 discusses techniques of directly manipulating implicit PDE objects with constraints to construct more flexible topological shapes, such as sketch sculpting and local region manipulations. We outline the system implementation in Section 6.

## 2. Related work

Different from traditional free-form spline-based modeling techniques, Bloor and Wilson [5] introduced a method that defines smooth surfaces as solutions of *elliptic* PDEs. Since its initial application on surface blending, the PDE approach has broadened its applications for free-form surface design, solid modeling, and interactive surface editing [6,7,42] during the past decade. In principle, the PDE-based method has the advantage that most of the information defining an object comes from its boundaries. This permits an object to be generated and controlled by a very few parameters such as boundary-value conditions and global coefficients associated with an elliptic PDE. This

PDE technique was then used for modeling parametric surfaces and solids with global geometric features. To obtain interactive sculpting and local control, we [14,15] proposed an integrated model which combined PDE surfaces and physics-based modeling techniques to offer users direct manipulation for the PDE surfaces with generalized boundary constraints and user-specified features. We [16] extended the PDE technique's coverage from surfaces to solids in order to provide users a set of direct editing toolkits to model the real-world objects with interior material distribution. Zhang and You [45] investigated three different orders, i.e. second, mixed, and fourth-order of PDEs as surface representation techniques and demonstrated the use and effectiveness of the PDE method for free-form surface design.

However, because the aforementioned PDE methods define objects over the regular parametric domain, they (like other parametric representation techniques) have limitations in handling arbitrary topological shapes, which can be easily achieved by implicit functions.

Implicit functions offer a different way for shape modeling by using certain scalar field functions to define geometric entities. In the past several years, implicit functions have been widely developed as a powerful design and manipulation tool for graphical models. In 1994, Witkin and Heckbert [44] introduced an approach using particles to sample and control implicit surfaces. A set of particles are locked onto a surface and act as *control points* for the implicit surface. The surface shape can be manipulated by moving particles interactively. Ferley et al. [18] presented a *sculpture metaphor* for rapid shape prototyping. In their approach, the sculpted shape is defined as the iso-surface of a spatially sampled scalar field and can be manipulated by adding, removing, painting, or smoothing material and applying free-form and stamp tools. These techniques only provide interactive and practical sculpting tools for implicit surfaces.

As for implicit solids, Savchenko et al. [34] introduced a novel approach for the reconstruction of geometric models from given point sets using volume splines. Raviv and Elber [32] presented an interactive sculpting technique that uses the zero level-set of the scalar, tensor-product, uniform trivariate B-spline functions to represent 3D objects. The trivariate functions have a control volume that consists of scalar control coefficients. Users can indirectly sculpt the object by modifying relevant scalar control coefficients of the trivariate B-spline functions in different levels of details. Hua and Qin [23,24] developed interactive solid sculpting toolkits with haptics on implicit B-spline solids defined through the use of B-spline control coefficients over the intensity field. However, the control of B-spline coefficients is less intuitive to ordinary users in general.

Implicit functions can also be used for shape reconstruction and 3D morphing process. Turk and O'Brien [40] used variational implicit functions to achieve shape morphing and surface reconstruction. They employed the radial basis function (RBF) method to construct an implicit function, which interpolates the given dataset and minimizes the thin-plate energy. Yet since the RBF method is a global variational interpolating approach, any changes in the dataset will cause recalculation of the entire system. It's time-consuming for direct manipulation and not applicable for local sculpting of complex implicit models. Ohtake et al. [29] presented a *multi-level partition of unity* implicit surface supporting local features, but it is sensitive to the quality of input data.

Level-set method is another popular technique to model implicit objects. Zhao et al. [47] proposed a *weighted* minimal surface model based on variational formulations and PDE techniques to construct a surface from scattered data. They used the level-set method as a numerical technique to evolve the implicit surface continuously following the gradient descent of the energy functional for the final reconstruction. Their level-set model is governed by a time evolution PDE with velocity at the level-sets given by the motion equation of the original surface. The level-set method is based on a continuous formulation using PDEs and deforms an implicit surface according to various equations of motion depending on geometry, external forces, or certain energy minimization. It can easily handle topological changes and reduce noises in the dataset. The level-set method mainly focuses on implicit objects reconstructed from scattered datasets. Problems for interpolating curve sketches, especially open curve sketches have not been addressed. The shape deformation using the level-set method is often obtained by manipulating the speed functions in the level-set formulation [1,28], which is non-intuitive for general users.

Despite the modeling advantages of implicit functions, there are still difficulties for intuitive design and direct manipulation of implicit surfaces and solids in general. We integrate the implicit functions with the parametric PDE to offer users modeling advantages of both types of techniques. Instead of time evolution PDEs used in the level-set method, we employ static elliptic PDEs for boundary value problems. In particular, we introduce a novel technique which defines volumetric implicit objects as solutions of the elliptic PDEs of scalar intensity fields under *generalized* boundary constraints, including sketch curves, scattered data points, as well as volumetric datasets. The constraints may be associated by different intensity values, which offers more degrees of freedom than previous implicit techniques. Our implicit PDE model can be used for geometric shape design, object reconstruction, damaged data recovery, and shape blending. Implicit PDE objects can be manipulated by modifying the initial constraints or directly changing intensity values in the interior of the volumetric space. The implicit PDE method recovers not only the target object, but also the entire working space by the given information. Our system does not require the constraints to be closed datasets, which provides modeling potentials for open surfaces. To visualize implicit objects of scalar

intensity field, we can either use the Marching Cube method [25] which calculates the triangulated iso-surface at a selected intensity value on discretized sampling grids, or output the volumetric data in the working space to other volume rendering systems such as Pov-Ray or Vol-Vis systems.

## 3. Formulating implicit PDEs

### 3.1. Implicit elliptic PDE formulation

The implicit PDEs employed in this paper are founded upon the parametric PDE solid models [16]. In order to take advantage of the interactive feature associated with the parametric PDE modeling techniques, we use elliptic PDEs to define scalar intensity field for modeling implicit objects. Because higher-order PDEs can provide higher-order continuity for the scalar intensity value distribution, we employ a fourth-order elliptic PDE to model the scalar field for smooth results with tangential continuity, especially when dealing with shape blending and damage data recovery in which most of information are specified as constraints.

In particular, we formulate the unknown function as the intensity field function $d(x, y, z)$ defined in the 3D physical space of $x, y$, and $z$. The corresponding implicit PDE is formulated as follows:

$$\left(a^2\frac{\partial^2}{\partial x^2}+b^2\frac{\partial^2}{\partial y^2}+c^2\frac{\partial^2}{\partial z^2}\right)^2 d(x,y,z)=0, \tag{1}$$

where $x, y$, and $z$ are coordinate variables of 3D physical space varying from 0 to 1, respectively, which form a unit cube as the working space; $a, b$, and $c$ are arbitrary blending coefficient functions of $x, y, z$ defining material properties of the implicit space, which are initially defined as constants throughout the entire working space. The blending coefficient functions control the relative intensity blending and the level of variable dependence among $x, y$, and $z$ directions. For example, according to Eq. (1), if $a$ is given as a large value for all sampling points in the working space, then the contributions of $d(x, y, z)$ along $x$ direction will be relatively small in comparison with the other two directions. Hence, the coefficient functions will affect the solution of Eq. (1).

Because the numerical techniques used in this paper to solve the fourth-order elliptic PDE are suitable for other boundary value PDEs, we also incorporate a second-order PDE into our system:

$$\left(a^2\frac{\partial^2}{\partial x^2}+b^2\frac{\partial^2}{\partial y^2}+c^2\frac{\partial^2}{\partial z^2}\right)d(x,y,z)=0, \tag{2}$$

which is less time-consuming to solve with less continuous intensity distribution and can be used for initial guess of intensity values of the objects.

Because $a(x, y, z), b(x, y, z)$, and $c(x, y, z)$ are allowed to vary across $d(x, y, z)$, i.e. different locations in the physical domain may have different smoothing coefficient values, local control on implicit PDE objects can be easily achieved.

To obtain direct and local manipulations on the implicit PDE objects, we solve Eqs. (1) and (2) using numerical methods based on finite-difference approximations of the PDEs, which require at least six boundary conditions at $x = 0, x = 1, y = 0, y = 1, z = 0, z = 1$ defining the intensity values at three boundary surface pairs of the 3D working space in order to derive a unique solution. However, in most applications, there are no such boundary conditions available for modeling implicit objects, especially in the case of using implicit functions for shape reconstruction, where the constraints are usually defined by certain contouring sketch curves or scattered points assigned with specified intensity values inside the 3D working space. In such cases, the intensity distributions on the boundaries are unknown. Thus, such problems cannot be solved by traditional finite-difference methods directly. However, we may approximate the intensity distribution for this type of problems as follows. First, we find an initial guess of the volumetric working space using certain techniques. Second, we use the guessed boundary values as boundary conditions, and enforce additional constraints according to the original data. Third, we perform iterative finite-difference techniques to get an approximated solution for the entire working space based on these constraints. After that, direct manipulations and local sculpting inside the working space can be enforced by adding additional constraints to the PDEs. Variational interpolating approaches are good candidates for shape reconstruction from scattered points, such as the RBF method [26,40] which creates a 3D implicit function to give an approximation interpolating the given constraints by minimizing certain energy functionals. We employ the RBF method to compute the initial guess of the implicit PDE objects defined by sketch curves. We can also calculate the intensity values on sampling grids using their distance to the constraints, because the implicit objects can be defined by distance functions. The algorithm we use to compute the distance field is the fast-tagging approach proposed by Zhao et al. [46]. Note that, because our goal is to obtain an initial guess of the working space according to constraints, there are other techniques which can provide satisfactory results.

### 3.2. Radial basis function

RBF is commonly used for scattered data interpolation, which is to generate a smooth surface that passes through a given set of scattered points. Scattered data interpolation sometimes can also be addressed using variational analysis where the desired solution is a function, $f(\vec{x})$, which minimizes certain energy functionals. In principle, the energy functional measures the quality of interpolation

subject to the interpolatory constraints $f(\vec{c}_i) = h_i$. It can be solved by a weighted sum of certain RBFs (note that, we use $\phi(\vec{x}) = |\vec{x}|^3$ in this paper). Then the interpolation function can be formulated as:

$$f(\vec{x}) = \sum_{i=1}^{n} w_i \phi(\vec{x} - \vec{c}_i) + P(\vec{x}), \tag{3}$$

where $\vec{c}_i$'s are coordinate vectors of the constraints, $w_i$'s are weights, and $P(\vec{x})$ is a polynomial only consisting of the linear and constant portions of $f$. According to the properties of the appropriate RBFs, the interpolation function minimizes the thin-plate energy while satisfying the data interpolation requirement. By applying the constraints to Eq. (3), we can obtain a linear equation system whose unknowns are the weights and coefficients of the polynomial $P$. This system can be solved using standard solvers of linear equations.

However, the RBF method requires gradient information of the datasets, and time and space complexity of the equation system depends on the number of constraints, so it is not suitable for reconstruction and interactive sculpting of large scattered datasets with arbitrary constraints. Because our goal here is to simply make an initial guess for our implicit PDE shape, distance approximation techniques such as fast-tagging algorithm which computes the signed distance field of the working space according the constraints can give satisfactory results for such input.

### 3.3. Numerical simulation

In order to easily enforce additional constraints for direct manipulations of implicit objects, we resort to numerical techniques based on the finite-difference approximation and iterative methods for linear equations to solve the implicit PDEs with predefined boundary values or approximated initial guess from sketch curves/scattered points. The iterative methods will arrive at an approximated solution with user-specified error tolerances. Numerical algorithms also facilitate the material modeling of anisotropic distribution. A multi-grid-like iterative solver is used to improve the system performance.

The finite-difference method divides the working space into discrete grids along $x, y, z$ directions and transforms a continuous PDE into a set of simultaneous algebraic equations by sampling the partial derivatives in the equation for each grid point with their finite-difference approximations. The algebraic equation system can be solved numerically either through a direct procedure or an iterative process for an approximated solution of the continuous PDE.

Based on Taylor's expansion, the derivatives of a univariate function can be approximated using the central-difference scheme $f'(x) = (f(x+h) - f(x-h)/2h, f''(x) = [f(x+h) - 2f(x) + f(x-h)]/h^2$, where $h$ denotes the spatial interval along $x$ direction. This can be generalized to all partial derivatives on trivariate implicit geometry,

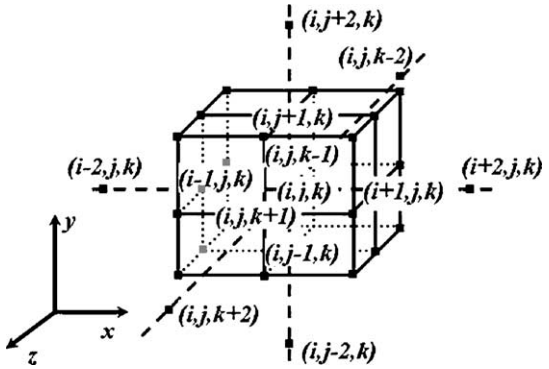


Fig. 1. The point discretization of an implicit function.

by dividing $x, y, z$ domain into $l, m$, and $n$ discretized grids, respectively. We represent the function $d(x, y, z)$ by its values at the discrete set of points $(x_i = i\Delta x, y_j = j\Delta y, z_k = k\Delta z)$, $i = 0, 1, \ldots, l-1$, $j = 0, 1, \ldots, m-1$, and $k = 0, 1, \ldots, n-1$. $\Delta x, \Delta y, \Delta z$ are the grid spacing along $x, y, z$ directions. We write $d_{i,j,k}$ for $d(x_i, y_j, z_k)$ and $\{i, j, k\}$ for grid point $(x_i, y_j, z_k)$ for sake of simplicity (Fig. 1). We use finite-difference representations of second-order and fourth-order partial derivatives $(\partial^2 d_{i,j,k})/(\partial x^2)$, $(\partial^4 d_{i,j,k})/(\partial x^4)$, and $(\partial^4 d_{i,j,k})/(\partial x^2 \partial y^2)$ at $\{i, j, k\}$ as examples:

$$\frac{\partial^2 d_{i,j,k}}{\partial x^2} = \frac{d_{i-1,j,k} + d_{i+1,j,k} - 2d_{i,j,k}}{(\Delta x)^2},$$

$$\frac{\partial^4 d_{i,j,k}}{\partial x^4} = \frac{d_{i-2,j,k} + d_{i+2,j,k} - 4d_{i-1,j,k} - 4d_{i+1,j,k} + 6d_{i,j,k}}{(\Delta x)^4},$$

$$\frac{\partial^4 d_{i,j,k}}{\partial x^2 \partial y^2} = \frac{d_{i-1,j-1,k} + d_{i-1,j+1,k} + d_{i+1,j-1,k} + d_{i+1,j+1,k}}{(\Delta x)^2 (\Delta y)^2} + \frac{-2d_{i-1,j,k} - 2d_{i+1,j,k} - 2d_{i,j-1,k} - 2d_{i,j+1,k} + 4d_{i,j,k}}{(\Delta x)^2 (\Delta y)^2}.$$

Other partial derivatives along $y$ and $z$ directions can be computed similarly.

Substituting partial derivatives by finite-difference representations at grid points, Eq. (1) can be rewritten as:

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

where $\mathbf{A}$ represents the discretized differential operator in $(l \times m \times n) \times (l \times m \times n)$ matrix form, and each row in $\mathbf{A}$ consists of coefficients of the difference equation for the corresponding grid point. $\mathbf{A}$ is also controlled by the blending functions $a(x, y, z), b(x, y, z)$, and $c(x, y, z)$. D collects the unknown intensity values at the grid points, and $\mathbf{b}$ is defined by the value of constraints:

$$\mathbf{A} = [\mathbf{A}_{(0,0,0)}, \mathbf{A}_{(0,0,1)}, \ldots, \mathbf{A}_{(l-1,m-1,n-1)}]^T,$$

$$\mathbf{A}_{(i,j,k)} = [A_{(i,j,k),(0,0,0)}, \ldots, A_{(i,j,k),(l-1,m-1,n-1)}],$$

$$\mathbf{D} = [d_{(0,0,0)}, d_{(0,0,1)}, \ldots, d_{(l-1,m-1,n-1)}]^T,$$

$$\mathbf{b} = [b_{(0,0,0)}, b_{(0,0,1)}, \ldots, b_{(l-1,m-1,n-1)}]^T.$$

**A** and **b** are defined as follows: given a grid point $\{i,j,k\}$, let its index $d = i \times l \times m + j \times m + k$ be represented as $(i,j,k)$. If it is a constraint point, all elements in $\mathbf{A}_{(i,j,k)}$ have value 0 except $A_{(i,j,k),(i,j,k)} = 1$, and $b_{(i,j,k)}$ is set to be the intensity value defined by the constraint. If it is free, the value of $A_{(i,j,k),(i',j',k')}$ depends on contribution of $\{i',j',k'\}$ in the difference equation at $\{i,j,k\}$, and $b_{(i,j,k)} = 0$. Fig. 1 shows the grid points contributing for $\{i,j,k\}$ in the $d$th row of **A**, i.e. $\mathbf{A}_{(i,j,k)}$. All the values of $A_{(i,j,k),(i',j',k')}$ are set to be 0 except:

$$A_{(i,j,k),(i,j,k)} = 6\left(\frac{a_{i,j,k}^4}{\Delta x^4} + \frac{b_{i,j,k}^4}{\Delta y^4} + \frac{c_{i,j,k}^4}{\Delta z^4}\right) + 8\left(\frac{a_{i,j,k}^2 b_{i,j,k}^2}{\Delta x^2 \Delta y^2} + \frac{a_{i,j,k}^2 c_{i,j,k}^2}{\Delta x^2 \Delta z^2} + \frac{b_{i,j,k}^2 c_{i,j,k}^2}{\Delta y^2 \Delta z^2}\right),$$

$$A_{(i,j,k),(i\pm1,j,k)} = -4\left(\frac{a_{i,j,k}^4}{\Delta x^4} + \frac{a_{i,j,k}^2 b_{i,j,k}^2}{\Delta x^2 \Delta y^2} + \frac{a_{i,j,k}^2 c_{i,j,k}^2}{\Delta x^2 \Delta z^2}\right),$$

$$A_{(i,j,k),(i,j\pm1,k)} = -4\left(\frac{b_{i,j,k}^4}{\Delta y^4} + \frac{a_{i,j,k}^2 b_{i,j,k}^2}{\Delta x^2 \Delta y^2} + \frac{b_{i,j,k}^2 c_{i,j,k}^2}{\Delta y^2 \Delta z^2}\right),$$

$$A_{(i,j,k),(i,j,k\pm1)} = -4\left(\frac{c_{i,j,k}^4}{\Delta z^4} + \frac{a_{i,j,k}^2 c_{i,j,k}^2}{\Delta x^2 \Delta z^2} + \frac{b_{i,j,k}^2 c_{i,j,k}^2}{\Delta y^2 \Delta z^2}\right),$$

$$A_{(i,j,k),(i\pm2,j,k)} = \frac{a_{i,j,k}^4}{\Delta x^4},$$

$$A_{(i,j,k),(i,j\pm2,k)} = \frac{b_{i,j,k}^4}{\Delta y^4},$$

$$A_{(i,j,k),(i,j,k\pm2)} = \frac{c_{i,j,k}^4}{\Delta z^4},$$

$$A_{(i,j,k),(i\pm1,j\pm1,k)} = \frac{2a_{i,j,k}^2 b_{i,j,k}^2}{\Delta x^2 \Delta y^2},$$

$$A_{(i,j,k),(i\pm1,j,k\pm1)} = \frac{2a_{i,j,k}^2 c_{i,j,k}^2}{\Delta x^2 \Delta z^2},$$

$$A_{(i,j,k),(i,j\pm1,k\pm1)} = \frac{2b_{i,j,k}^2 c_{i,j,k}^2}{\Delta y^2 \Delta z^2}.$$

The matrix **A** is called 'tridiagonal with fringes' [31].

Similarly, using finite-difference techniques, Eq. (2) can be rewritten as:

$$\mathbf{A}'\mathbf{D} = \mathbf{b}', \tag{5}$$

Our implicit PDE is open along all of $x$, $y$, and $z$ directions, so forward/backward difference approximations shall be utilized when computing partial derivatives near the six boundaries instead. Arbitrary boundary and additional constraints can be easily enforced by the finite-difference method. In our system, after making the initial guess of the intensity values, we fix the intensity values at boundaries, so that the manipulations on the implicit objects can be performed using the finite-difference iterative solver. In general, this type of elliptic PDEs allows designers to choose (various) constraints based on diverse design tasks.

### 3.4. Constrained system

One attractive advantage of the PDE modeling techniques is that the interior of the objects is controlled by PDEs without the need of extra specification for interior material distribution. More importantly, users can modify an implicit PDE object by enforcing additional hard constraints of desired intensity values anywhere inside the working space without violating previously defined conditions. Additional hard constraints inside the working space introduce a set of new equations into the system to replace the corresponding original difference equations. For example, if we want to set the intensity value $d_{i,j,k}$ as a particular constant value $d_0$, the equation $d_{i,j,k} = d_0$ will be used to replace the discretized difference equation approximating the PDE at the point $\{i,j,k,\}$, i.e. $A_{(i,j,k)(i,j,k)} = 1$, all other $A_{(i,j,k)(i',j',k')} = 0$, and $b_{(i,j,k)} = d_0$. After replacing all the equations according to the constraints, Eq. (4) becomes

$$\mathbf{A}_c D = \mathbf{b}_c, \tag{6}$$

where $\mathbf{A}_c$ and $\mathbf{b}_c$ are obtained by replacing $k(k > 0)$ equations in the original system with those derived from additional $k$ constraints at the corresponding coordinate positions. The constrained system for the second-order Eq. (5) has the similar form:

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

### 3.5. Iterative method

With boundary conditions, we solve the linear Eqs. (4)–(7) using finite-difference-based iterative techniques. These methods make immediate use of the sparse matrix structure on the left-hand side of the equations. Using the matrix **A** in Eq. (4) as an example, **A** is split into two parts

$$\mathbf{A} = \mathbf{A}_d - \mathbf{A}_r, \tag{8}$$

where $\mathbf{A}_d$ consists of the diagonal elements of **A** and zeros elsewhere, and $\mathbf{A}_r$ is the remainder. Then Eq. (4) becomes

$$\mathbf{A}_d\mathbf{D} = \mathbf{A}_r\mathbf{D} + \mathbf{b}. \tag{9}$$

The iterative methods start from choosing an initial guess $\mathbf{D}^{(0)}$ and then solving successively by iterating $\mathbf{D}^{(s)}$ from

$$\mathbf{A}_d\mathbf{D}^{(s)} = \mathbf{A}_r\mathbf{D}^{(s-1)} + \mathbf{b}. \tag{10}$$

The same idea can be applied to Eqs. (5)–(7).

In the case of predefined boundary conditions, we compute the initial guess using simple linear interpolations based on the constraints. The iteration will stop at $\mathbf{D}^{(s)}$ for an approximated solution when the difference between $\mathbf{D}^{(s)}$ and

$\mathbf{D}^{(s-1)}$ is less than a threshold (we use $10^{-9}$ in this paper). Certain variants of iterative techniques exist for solving the aforementioned linear equations [38]. In this paper, we employ the Gauss-Seidel iteration, which uses the updated value of the iteration result at a grid point on the right-hand side of Eq. (10) as soon as it becomes available. To further speed up the converging rate of Gauss-Seidel iteration, we take into account the error factor, which is characterized by the difference between the approximation and the real solution. This leads to the method of successive over-relaxation iteration, or SOR iteration. Nonetheless, the discretization of volumetric implicit PDE space results in a very large number of linear equations. This causes the slow convergence of iterative methods. To achieve a solution faster, we start solving the equations at a coarse grid with down-sampled constraints and interpolate the solution at finer grids to compute the initial guess for the iterative methods at the finer resolution. The convergent rate of the iterative solvers can be greatly increased.

## 4. Boundary conditions for different applications

To construct an implicit PDE object, first we need to outline the rough shape of the object, which can be defined through boundary conditions or special constraints such as curve contours and scattered data points in the working space that the object interpolates. The form of boundary constraints varies for different applications. Our implicit PDE techniques accept boundary conditions for applications such as shape blending, object recovery, and shape reconstruction from sketch curves and scattered data points. Fig. 2 illustrates different types of boundary conditions in simplified 2D cases.

### 4.1. Shape design using traditional boundary constraints

The implicit PDE techniques can model geometric shapes by computing the information of the whole working space based on traditional boundary constraints with optional cross-sectional details inside the working space. Such boundary conditions are defined as intensity values sampled at certain resolution from input or use some analytic functions to generate implicit boundary functions $d(0, y, z)$, $d(1, y, z)$, $d(x, 0, z)$, $d(x, 1, z)$, $d(x, y, 0)$, $d(x, y, 1)$ and a collection of cross-sectional scalar intensity functions $d(x_i, y, z)$, $d(x, y_j, z)$, or $d(x, y, z_k)$, where $x_i, y_j, z_k \in (0, 1)$ are constants. These functions are sampled at specified resolution to provide a set of intensity values inside the working space. Using these values as generalized boundary conditions, we introduce certain number of new equations and the linear equation system has the form of Eqs. (6) or (7) which can be solved using above mentioned techniques. Fig. 3 shows examples of the fourth-order and second-order PDEs, respectively. Although Eq. (6) takes more time to solve, it provides higher-order continuity of intensity

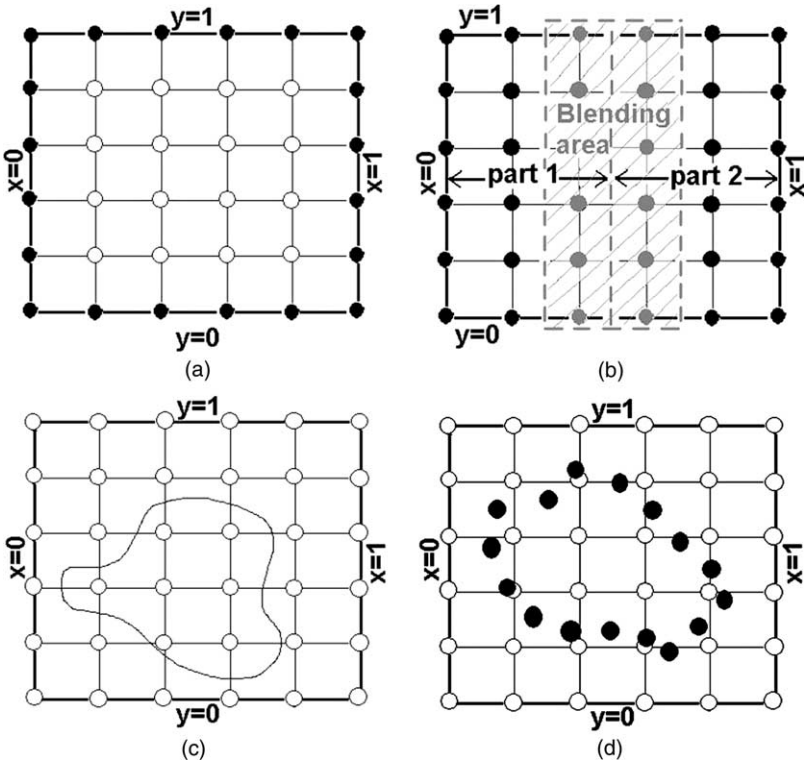


Fig. 2. 2D illustrations of different types of boundary conditions. (a) Traditional boundary constraints; (b) boundary conditions for shape blending; (c) sketch-curve constraints; (d) scattered-point constraints.

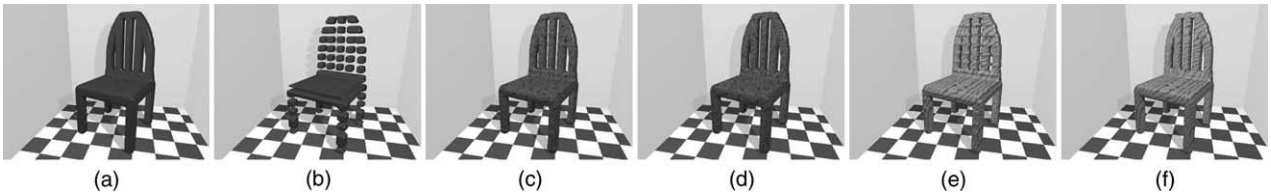


Fig. 3. Examples of implicit PDE objects generated from cross-sectional boundary conditions. (a) Original object rendered by POV-RAY; (b) cross-sectional boundary conditions by removing several data slices along the $y$-direction from the original data; (c) and (d) are recovered fourth-order implicit PDE objects from (b) by solving Eq. (1) with $b = 1.2$ and 4.8, respectively; (e) and (f) are corresponding second-order objects. The fourth-order PDE provides more continuous results.

distributions in the working space in comparison with the results from Eq. (7).

### 4.2. Shape blending

Our PDE formulations define the interior information of implicit objects via differential properties, which means that it is possible to automatically recover the missing information from partial data using our prototype system and guarantee intensity continuity of non-constrained parts of the working space. This feature can be applied to shape blending process by placing the objects to be blended into the working space and the system will compute the connecting parts between those objects. Such kind of datasets form another type of initialization with pre-defined boundary constraints, which gives most of the information with only a small portion of the working space missing. The missing information of the working space can be approximated based on the remaining part using our PDE formulations. An example of shape blending is shown in Fig. 4 including blended results using different order PDEs, where the fourth-order blended shape is smoother than the second-order result. The above two types of boundary conditions allow our system to model volumetric datasets.

### 4.3. Shape reconstruction from sketch curves

To maximize the modeling potential of implicit PDEs, we develop a set of toolkits using PDE techniques to reconstruct objects from spatial sketch curves of specified intensity values. Because with this type of constraints, the boundary information around the working space is missing, it is extremely difficult to directly solve the implicit PDE under such constraints. Therefore, we employ techniques such as the RBF method for the interpolation problems to obtain an initial guess for the implicit PDE shapes subject to sketch curve constraints. We then use the iterative solver to get a smooth solution. When performing the RBF method, the gradient information indicating the change of the intensity values around the constraints will be needed to define the inside and the outside of the reconstructed shape. If the gradient information is not provided by users, our system calculates the gradient at each sample point of the constraints according to the normal of the local tangent plane of the curve at that point, as explained in Fig. 5. Our system also allows designers to interactively input certain sketch curves such as B-spline curves with specified intensity values, which permits the initial sketch curves being modified directly. Note that, the sketch curves are not required to be planar curves. Moreover, they can even be open curves, which may result in open iso-surfaces instead of solid objects. Fig. 6 shows examples obtained from sketch curves.

When modeling more complex shapes from sketches, usually there are a large number of sketch curves to be enforced, which will increase the number of calculations dramatically. Moreover, sometimes the sketch curves are only designed to model the local area they resides, so their global contribution are not desirable. To address such issues, our system allows users to compute the initial guess of implicit PDE objects using the RBF method for selected subset of sketch curves at any local region of the working domain without disturbing the outside areas. At the initialization stage, when using RBF method to compute the initial guess of the implicit shape, users are prompted to select interested curves, define the region in the working

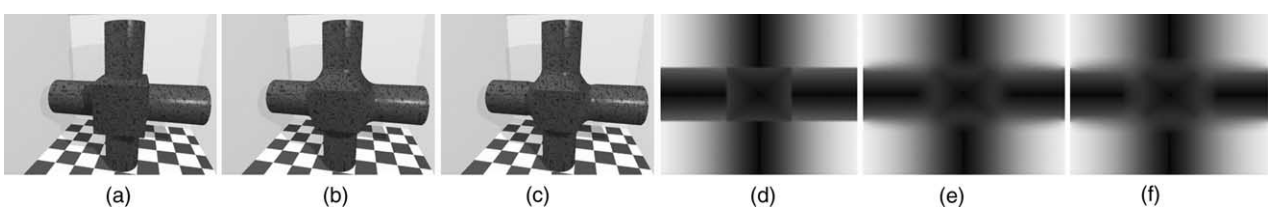


Fig. 4. Shape blending using implicit PDEs. (a) Original dataset shown in iso-surface; (b) blended object from (a) using the fourth-order PDE; (c) blended object using the second-order PDE; (d)–(f) are cross-section views of the working space for (a)–(c), respectively, where darkness increases with intensity.

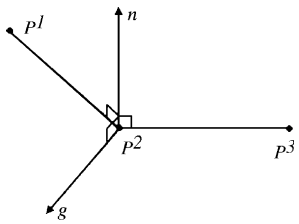


Fig. 5. Illustration of computing the gradient direction. $p1, p2$, and $p3$ are neighboring points on a discretized curve; $\vec{n}$ is the normal of their local plane; and $\vec{g}$ is the gradient vector for $p2$.

space to reconstruct the subset of the object, as well as to indicate if curves that only part of them inside the specified area can make contribution to the reconstruction. After all the sampled intensity values in each of the sub-regions of the working space are computed, our system can perform a global blending process to put sub-regions together. This feature can reduce the number of calculations of the RBF method, and provide fast reconstruction by sculpting sketch curves. Moreover, CSG sculpting tools can be easily enforced accordingly. Fig. 7 shows an example.

### 4.4. Shape reconstruction from unorganized scattered data points

Implicit functions are commonly used for shape reconstruction from scattered data points. In this paper, our implicit PDE model not only reconstructs objects from unorganized scattered data sets, but also recovers information of the entire working space where objects reside, with which direct manipulations of objects can be easily applied. Similar to the sketch curve constraints, intensity values at boundaries of the working space are unknown. However, for scattered points datasets where the number of constraints is extremely large and there is no gradient information available, RBF method is not suitable for computing the initial guess. In such case, we use the signed distance field approximation based on the constraints. The initial intensity value on the sampling grids are computed by the fast-tagging algorithm introduced by Zhao et al. [46] based on their signed distance to the data point constraints and we then use iterative solvers to conduct a smoothing task. Two examples are shown in Fig. 8.

## 5. Sculpting and manipulation toolkits for implicit PDEs

Our system provides a set of toolkits for global deformation and local editing of the implicit objects. Fig. 9 shows a snapshot of our prototype system while manipulating a selected sketch curve.

### 5.1. Modifying blending coefficients

The coefficient functions $a(x, y, z), b(x, y, z)$, and $c(x, y, z)$ can influence the solution of the implicit PDEs. They control the relative intensity blending and the level of variable dependence among $x$, $y$, and $z$ directions, thus they can be treated as generalized material properties over the volumetric working space. Consequently, users can control how the boundary and additional conditions influence the interior intensity distribution by modifying the length scale at arbitrary locations (i.e. $a_{i,j,k}, b_{i,j,k}$, and $c_{i,j,k}$). In general, users can

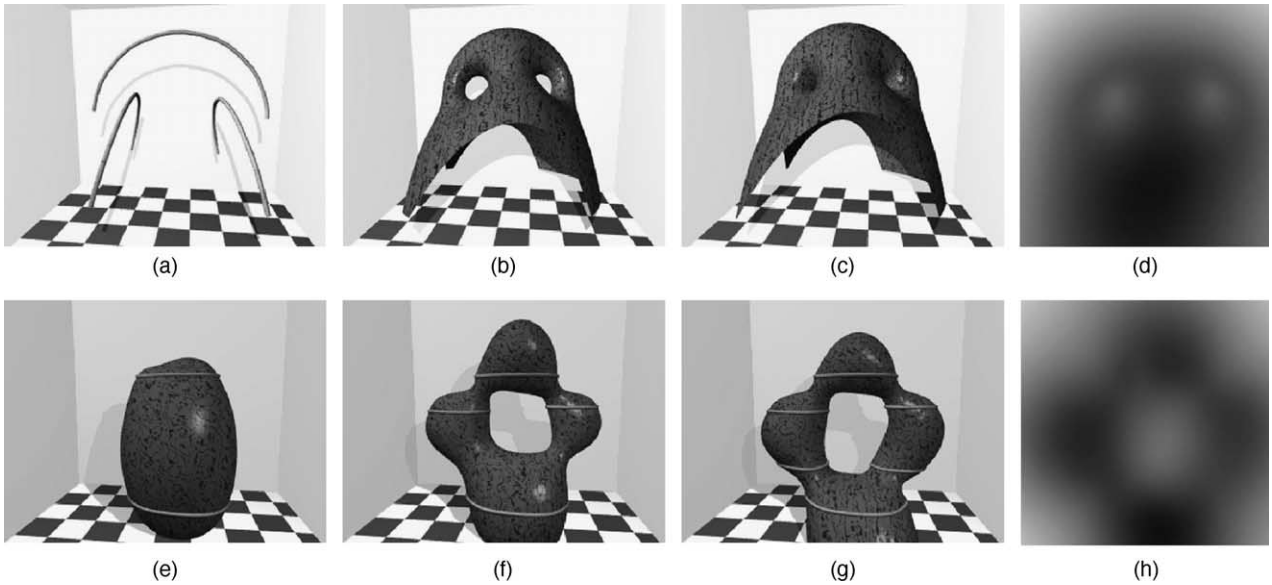


Fig. 6. Examples of shape reconstruction from sketch curves. (a) is a set of open curves without specified gradient information; (b) and (c) are iso-surfaces at different iso-values, respectively; (d) is a cross-section view of the implicit shape; (e)–(g) show an example of generating implicit shapes by incrementally defining a set of B-spline curves; (e) is an object defined by two curves; (f) is the refined object by adding two additional sketch curves; (g) is the shape reconstructed from six B-spline sketch curves; and (h) is a cross-section view.

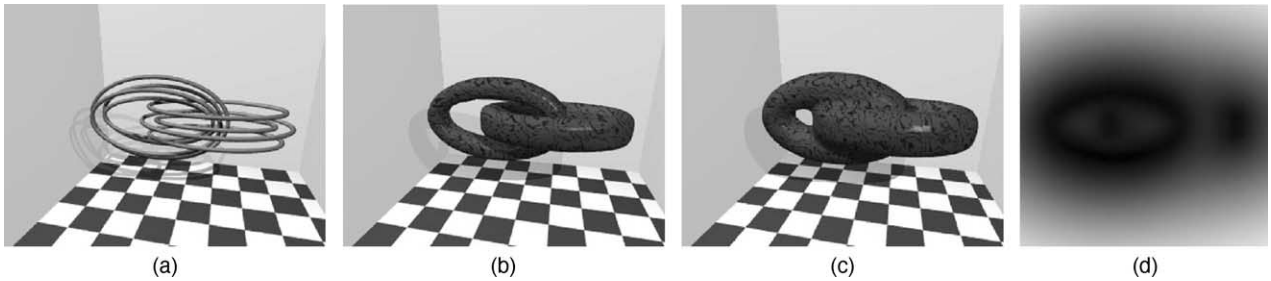


Fig. 7. Example for performing the RBF initialization locally. (a) Two set of sketch curves; (b) and (c) are reconstructed implicit shapes rendered at different iso-values; (d) a cross-section view.

define the control functions $a(x,y,z), b(x,y,z)$, and $c(x,y,z)$ interactively over the specified grid point $\{i,j,k\}$. Our system allows users to modify them locally to deform the shape. Fig. 3 has examples of implicit PDE objects subject to different coefficient values.

### 5.2. Sketch curve sculpting

Implicit objects can be defined by specifying a set of sketch curves which outline the rough shape of the objects. Our implicit PDE model provides interactive shape design toolkits to allow users to manipulate the sketch curves in order to deform the underlying reconstructed implicit object. The sketch curves defining the rough shape of the object can be obtained by either predefined curve network or B-spline curves from users' direct input. Our system allows users to modify the geometric shape, intensity value, as well as gradient directions of the sketch curves interactively in order to get the desired object.

In order to modify the sketch curves smoothly, B-spline approximations for those curves are calculated at the initialization stage, then users can sculpt the curves interactively by manipulating the B-spline control points via sculpting, translation, and rotation. Because the reconstructed implicit object is required to interpolate those sketch curves, which define its outlining shape approximately, it will follow the shape changes accordingly. Fig. 11 has an example of sculpting the shape of a selected sketch curve. The intensity values of sketch curves decide where the final shape of the implicit objects should pass through at the level-set of its value. By modifying the intensity values of selected curves, users can manipulate the objects accordingly. Furthermore, according to the gradient definition, the intensity values increase along gradient directions of sketch curves and decrease in the opposite directions in general. Gradient directions provide information of the intensity distributions starting at the sketch curves and propagating to the neighborhood, which

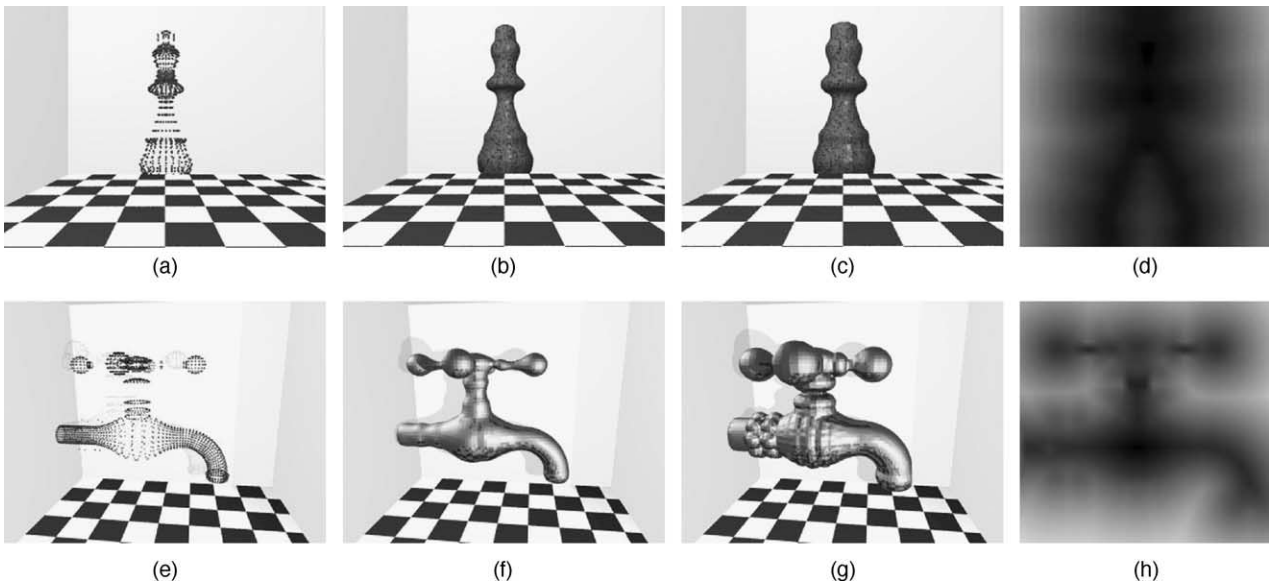


Fig. 8. Examples of shape reconstruction from scattered data points (b) and (c) are iso-surface at different intensity values of the object reconstructed from point set (a); (f) and (g) represent the reconstructed shape from dataset (e) at different iso-values; (d) and (h) are cross-section views.

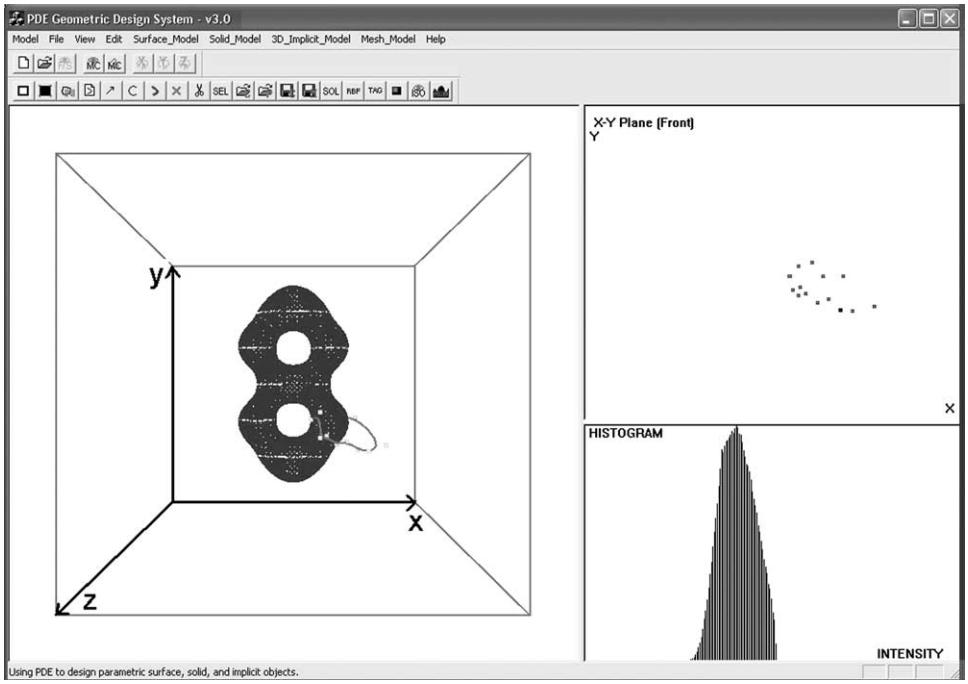


Fig. 9. Snapshot of the interface of our implicit PDE system.

defines the inside and outside of the object. Without the predefinition of gradient directions, the solution will be trivial. Therefore, gradient information of sketch curves is required for reconstructing a unique shape. Accordingly, changing the gradient directions at selected sketch curves means modifying directions of intensity changes in the implicit working space and will result in different implicit shapes. Our system allows users to specify the gradient direction of each individual sketch curve to construct different implicit PDE objects. Refer to Fig. 10 for examples of specifying and modifying gradient directions of the sketch curves. Without further specification, other examples in this paper have gradient directions pointing inward the curves by default.

### 5.3. Local manipulation of implicit PDE solids

Usually the sketch curve sculpting will deform the entire reconstructed shape, which only offers global manipulation and is less intuitive for ordinary users to handle. Even with the specification of local areas of interests containing the sculpted sketch curve, the sculpting will affect all the points in the selected regions. Moreover, sometimes the input constraints alone cannot guarantee a satisfactory solution of constructed shape. Therefore, direct modification in selected areas is desirable, especially when the overall recovered shape is satisfactory but minor changes in small localized areas are needed. Our system provides interactive tools for the intensity value modification in selected regions to sculpt the reconstructed shape. The modification will be enforced into Eqs. (6) or (7). Using the aforementioned techniques, we can solve Eqs. (6) and (7) to obtain the modified objects. Because for local manipulations, we only calculate the intensity updates in the neighborhood of selected regions, where the intensity values are governed through the PDE and the selected regions usually have relatively small number of grids comparing

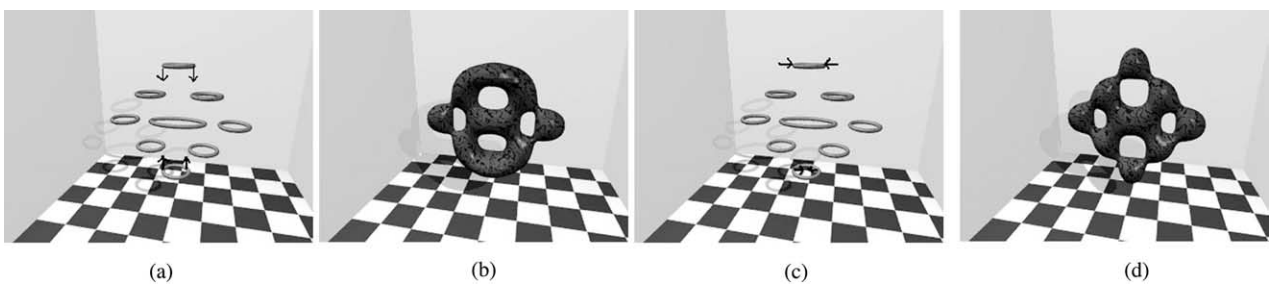


Fig. 10. Examples for specifying and changing gradient directions of sketch curves. (a) and (c) are two sets of curves with same geometric shape but different gradient directions, where the arrows show intensity increasing directions; (b) and (d) are corresponding implicit objects.

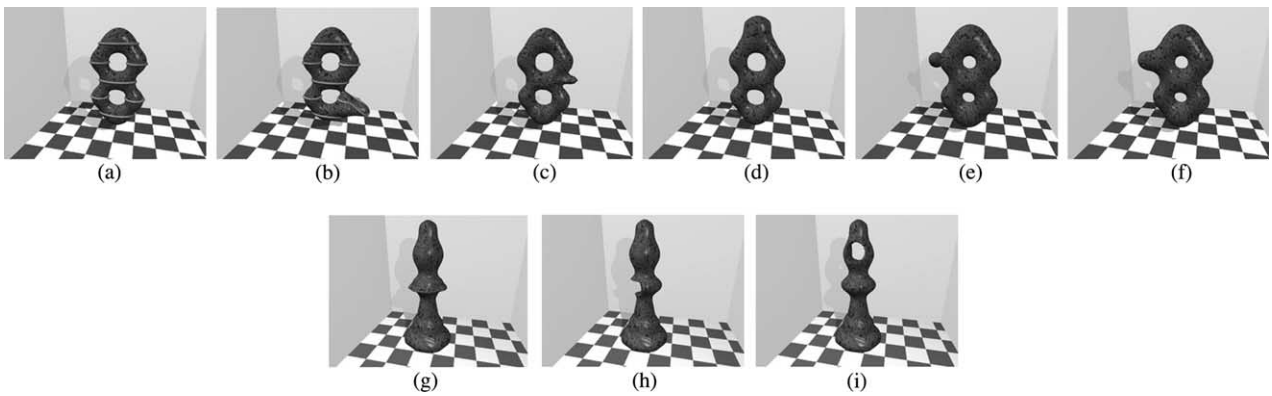


Fig. 11. Examples of enforcing curve and direct manipulation constraints. (a) Original object with sketch curves; (b) deformed object by sculpting a selected curve; (c) changing an iso-contour; (d) deformed object subject to local region constraints; (e) adding a sphere in the working space; and (f) is the corresponding deformed object subject to (e); (g) adding constraints for an object with sharp edges; (h) and (i) are two trimming examples.

with the entire working space, the update of the new intensity values for such regions will be quickly obtained through our finite-difference solver. Therefore, we can achieve interactive manipulations for local sculpting of implicit PDE objects.

Traditional implicit techniques for data reconstruction do not support direct manipulations on the arbitrary locations in the volumetric working space. The changes on the predefined constraints will cause global deformation. It is more desirable to offer users editing functionalities on the interior properties with interactive interface.

#### 5.3.1. Local intensity modification

Besides the local RBF approximation for local sketch curve sculpting, our system also allows users to specify any interior region of the sampling grids, and applies intensity changes only within the specified region. Alternatively, we can freeze the selected region and disallow any changes in the specified region. In our system, this can be done through interactively specifying the maximum and minimum sampling grid in $x, y$, and $z$ direction of the desired region in the sampling volumetric working space. Subsequently, any change within the region will have no influence on sampling points outside the region. The localized deformation can be easily achieved because only those equations corresponding to the points of the specified regions in Eq. (6) will be solved. In addition, the number of computations is reduced due to fewer number of equations involved in the local sculpting. In principle, all hard constraints can be viewed as some sort of local deformation. Fig. 11 shows examples of local deformation.

#### 5.3.2. Iso-surface sculpting

Users can also specify an iso-surface at a particular intensity value and use a cutting plane inside the volumetric working space to get a 2D iso-contour on the plane, then stretch, push, rotate the contour, as well as add desired intensity values at specified locations to modify the shape of the iso-surface and the intensity distribution of the interested areas. Refer to Fig. 11 for illustrative examples.

#### 5.3.3. CSG operations

We also offer several CSG sculpting tools such as using spheres and cubes to trim/extrude/sculpt implicit objects by adding more constraints on the sampling grids of the working space. This is extremely useful for such situations when there are some minor changes needed to be done in some local small regions. Such sculpting tools make our system compatible with CSG-based implicit models by treating those models as modeling tools. Examples are shown in Fig. 11.

#### 5.3.4. Gradient constraints

The intensity gradient $\nabla$ at a point $(x, y, z)$ in the intensity field can be defined as

$$\nabla d(x, y, z) = \left( \frac{\partial d(x, y, z)}{\partial x}, \frac{\partial d(x, y, z)}{\partial y}, \frac{\partial d(x, y, z)}{\partial z} \right).$$

By applying the finite-difference techniques, the gradient vector $\nabla d_{i,j,k}$ at a discretized grid point $\{i, j, k\}$ can be approximated as:

$$\left( \frac{d_{i+1,j,k} - d_{i-1,j,k}}{2\Delta x}, \frac{d_{i,j+1,k} - d_{i,j-1,k}}{2\Delta y}, \frac{d_{i,j,k+1} - d_{i,j,k-1}}{2\Delta z} \right).$$

It provides information about intensity changes in the neighborhood of $(x, y, z)$ in the working space. Therefore, changing the direction and length of the gradient vector of a selected grid point will affect the intensity distribution in its neighborhood, and as a result, deform the object. Our system allows users to pick a point inside the working space, specify the local region surrounding the point, and modify its gradient vector interactively, then the shape bounded by the specified local region will be deformed accordingly (refer to Figs. 12 and 13(a)).

#### 5.3.5. Curvature constraints

The mean curvature at point $(x, y, z)$ in the intensity field can be computed from the divergence of the intensity

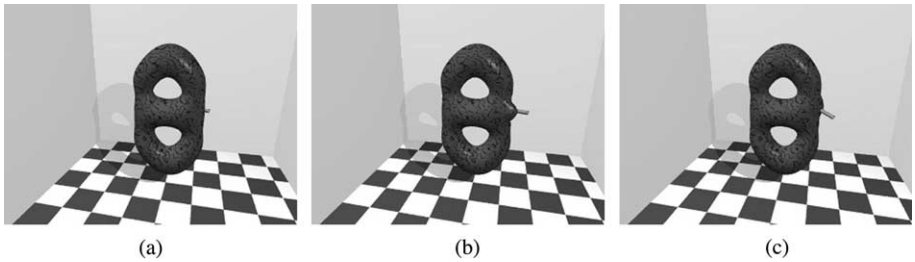


Fig. 12. Examples of enforcing gradient constraints. (a) Original object with the gradient vector at a selected point; (b) and (c) are deformed objects by changing the gradient at the point.

gradient of $(x, y, z)$, i.e. $\nabla\cdot\nabla d(x, y, z)$ [33]. In the discretized form, it can be approximated as $\nabla\cdot\nabla d_{i,j,k}$. Its definition is also related to the intensity value of the point's neighbors. By changing the curvature value at a point, the shape of the object will be changed. Our system allows users to manipulate the curvature at a selected grid point for implicit shape deformation. Fig. 13 shows examples.

## 6. Implementation and discussion

We develop a prototype software system that permits users to reconstruct geometric shapes defined by PDE-based implicit functions from a set of sketch curves, scattered data points, or volumetric datasets. Our system also allows interactive manipulation of reconstructed implicit PDE objects with various intensity constraints in the volumetric working space. The interactive sculpting of implicit PDE objects can be obtained via modification of predefined conditions and interior operations. The system is written in Visual C++ and runs on Windows95/98/NT/2000/XP. Fig. 14 illustrates the architecture of our modeling environment for implicit PDE objects. In particular, our system provides the following functionalities:

*Missing information recovery and shape blending.* The underlying implicit PDEs of our system provide a simple yet systematic mechanism to obtain the volumetric information satisfying specified constraints automatically. Such an advantage makes it possible to recover missing information of input datasets with our system. It can also be used to compute connecting parts between different objects in the working space which leads to shape blending.

*Shape reconstruction.* Users can interactively input and edit scattered data points or sketch curves with specified intensity values, then the system uses the RBF method or distance field approximation to calculate intensity values on the sampling grids within the volumetric working space as initial guesses for the iterative solver of the discretized implicit PDE to obtain approximated solutions for implicit PDE objects satisfying these conditions. Our system can model both close and open implicit shapes.

*Discrete models.* Our system supports implicit PDE objects obtained from solving the fourth-order and second-order elliptic PDEs using: (1) finite-difference discretization for the numerical solution of the elliptic PDEs in 3D working space; and (2) RBF approximation at arbitrary sub-regions in the working space for modeling localized details and performance speedup.

*Interactive and direct operations.* Users can also work directly on implicit PDE objects through: (1) local modification of blending coefficient functions; (2) sketch curve sculpting using B-spline manipulation; (3) gradient specification of selected curves; (4) local RBF approximation for improved time performance and interactive CSG manipulation; (5) interior deformation with additional constraints inside the working space; (6) iso-surface manipulation and direct manipulation of iso-contours at selected intensity values; and (7) gradient and curvature constraints inside the working space.

We employ iterative methods (e.g. Gauss-Seidel iteration) with multi-grid-like techniques to solve the implicit PDEs subject to various constraints. Besides original datasets or predefined sketch curves, our system allows users to interactively define and sculpt sketch

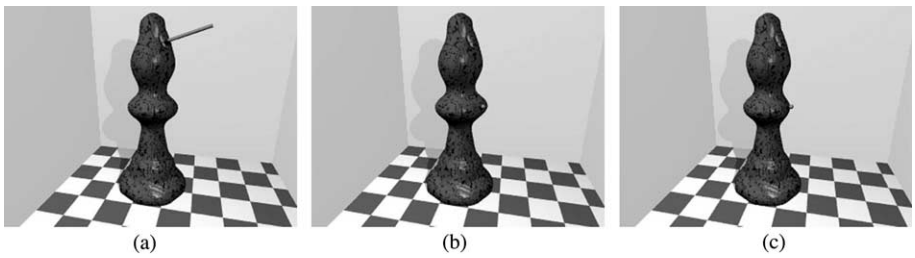


Fig. 13. Examples of enforcing gradient and curvature constraints. (a) Deformed object by changing gradient; (b) and (c) are deformed objects by changing curvature at a selected point (shown in green) from (a).

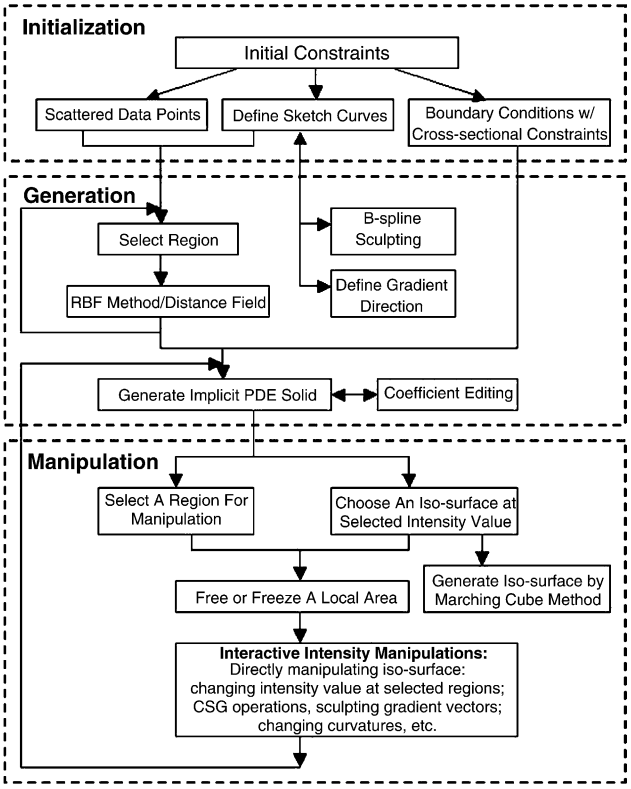


Fig. 14. System architecture and functionalities.

curves directly and specify gradients at selected curves. These constraints provide more freedom to designers and make intuitive design of implicit objects more cost-effective. We also enforce additional constraints directly inside the volumetric working space, apply local operations, and provide sculpting toolkits for the implicit objects, which facilitate the construction of implicit PDE objects of arbitrary topology. The PDE is solved by finite-difference techniques because they are simple, easy to implement, and suitable for complicated, flexible constraints. In general, the time and space complexity are increased with higher resolution as well as increased accuracy. Examples in this paper are rendered by POV-RAY.

Table 1 summarizes the numbers of constraints and CPU time of numerical solvers for the second-order and fourth-order implicit PDE examples when running on a Pentium 4.1.4 GHz PC. The resolution of the working space is $64 \times 64 \times 64$ for Fig. 3 and $65 \times 65 \times 65$ for other examples. The stopping threshold (difference between two iteration steps) is $10^{-9}$. 'Initial' stands for the initial guess where we use RBF method for sketch curve datasets and fast-tagging approximation for the scattered data points input. '2nd(s)' and '4th(s)' indicate the CPU time in seconds for solving the entire implicit second-order and fourth-order PDE working spaces based on the initial guess using multi-grid Gauss-Seidel iteration. The time performance of RBF and fast-tagging algorithms depends on the number of enforced constraints, while the convergent speeds of iterative methods are mainly determined by the sampling rates of the implicit working space.

Although the initialization of the implicit models are time-consuming because of the approximation of the entire working space, the local sculpting afterward will be interactive because only small number of sampling grids are involved. Table 2 summarizes the CPU time for the examples of our direct sculpting in local selected regions. 'Cons' stands for the number of constraints involved for the operation, 'Grids' represents the number of grid points in the selected region, and '4th(s)' gives the CPU time (seconds) for updating the intensity values in the selected area using the fourth-order PDE. The CPU time depends on the scale of the intensity change by the sculpting operation as well as the number of constraints and the size of the selected region. For instance, CSG operations usually enforce relatively larger intensity changes for constraints in selected regions than other operations such as gradient and curvature sculpting, hence they need more CPU time to update the region's intensity values.

Despite the direct and powerful modeling advantages of our PDE framework, the major difficulty associated with our PDE techniques is the convergent speed of finite-difference approximation for initial shapes. Thus, faster numerical approximation techniques for solving PDEs need to be considered to improve the time performance of our PDE modeling system.

Table 1
CPU time (seconds) of different solvers for several examples of implicit PDE objects with different number of constraints

| Examples | Constraints | Initial | 2nd(s) | 4th(s) |
|---|---|---|---|---|
| Fig. 3 | 169888 | N/A | 1.542 | 7.992 |
| Fig. 4 | 274086 | N/A | 3.04 | 13.7 |
| Fig. 6a | 180 | 5.889 | N/A | 379.766 |
| Fig. 6b | 720 | 18.872 | N/A | 416.312 |
| Fig. 8a | 1219 | 267.925 | N/A | 113.432 |
| Fig. 8e | 3154 | 359.657 | N/A | 148.283 |

Table 2
CPU time (seconds) of local direct manipulation examples of implicit PDE objects

| Examples | Cons | Grids | 4th(s) |
|---|---|---|---|
| Iso-contour Editing (Fig. 10c) | 8 | 1792 | 0.45 |
| Region Deformation (Fig. 10d) | 507 | 6358 | 3.17 |
| CSG-like Blending (Fig. 10f) | 108 | 1000 | 1.15 |
| Sharp-feature Creation (Fig. 10g) | 98 | 5046 | 0.23 |
| Cutting-1 (Fig. 10h) | 216 | 1000 | 0.82 |
| Cutting-2 (Fig. 10i) | 216 | 1000 | 0.82 |
| Gradient Sculpting (Fig. 12) | 7 | 294 | 0.09 |
| Curvature Manipulation (Fig. 13) | 7 | 294 | 0.09 |

## 7. Conclusion

We have unified the popular implicit function techniques with the powerful parametric PDE framework to demonstrate more modeling advantages of the PDE-based paradigm. Our prototype system supports interactive shape design of implicit PDE objects through global and local deformation of scattered data points or sketch curves. The implicit PDE model can be defined as the solution of the elliptic PDEs over a scalar intensity field with either scattered-point datasets or a set of sketch curves as generalized boundary and additional constraints. Our implicit PDE approach can also provide an approximation for the missing or blending part in the working space with most of the intensity information already known. Our software environment offers users a set of interactive and direct shape modeling toolkits including: sketch curve sculpting and gradient manipulation, intensity value modification in selected regions, gradient and curvature manipulations inside the working space, and iso-contour manipulation of specified intensity value inside the volumetric domain. These toolkits provide users an intuitive interface to model implicit PDE objects satisfying a set of design criteria and functional requirements. Our integrated approach and novel PDE techniques further expand the geometric coverage and the topological flexibility of the conventional PDE methodology to implicit functions, and forge ahead toward the realization of the full potential of PDE technology in shape modeling and other visual computing fields.

## Acknowledgements

This research was supported in part by the NSF ITR grant IIS-0082035, the NSF grant IIS-0097646, the NFS grant CCR-0328930, the NSF ITR grant IIS-0326388, Alfred P. Sloan Fellowship, and Honda Initiation Award.

## References

[1] Bærentzen A, Christensen N. Volume sculpting using level-set method. In Shape Modeling International 2002, Banff, Alberta, Canada; 2002. p. 175–82.
[2] Bertalmio M, Sapiro G, Caselles V, Ballester C. Image inpainting. In SIGGRAPH 2000, New Orleans, USA; 2000. p. 417–24.
[3] Bloomenthal J, Bajaj C, Blinn J, Cani-Gascuel M-P, Rockwood A, Wyvill B, Wyvill G. Introduction to Implicit Surfaces. Los Altos, CA: Morgan Kaufmann; 1997.
[4] Bloomenthal J, Wyvill B. Interactive techniques for implicit modeling. Comput Graphics 1990;24(2):109–16.
[5] Bloor MIG, Wilson MJ. Generating blend surfaces using partial differential equations. Comput Aided Des 1989;21(3):165–71.
[6] Bloor MIG, Wilson MJ. Using partial differential equations to generate free-form surfaces. Comput Aided Des 1990;22(4): 202–12.
[7] Bloor MIG, Wilson MJ. Functionality in solids obtained from partial differential equations. Computing Suppl 1993;8:21–42.
[8] Breen D, Whitaker R. A level-set approach for the metamorphosis of solid models. IEEE Transact Vis Comput Graphics 2001;7(2): 173–92.
[9] Carr J, Beatson R, Cherrie J, Mitchell T, Fright W, McCallum B. Reconstruction and representation of 3D objects with radial basis functions. In SIGGRAPH 2000, Los Angeles, USA; 2001. p. 67–76.
[10] Cohen-Or D, Levin D. Three-dimensional distance field metamorphosis. ACM Transact Graphics 1998;17(2):116–41.
[11] Cutler B, Dorsey J, McMillan L, Müller M, Jagnow R. A procedural approach to authoring solid models. In SIGGRAPH 2002, San Antonio, TX; 2002. 302–11.
[12] Desbrun M, Cani-Gascuel M-P. Active implicit surfaces for animation. Graphics Interface 1998;143–50.
[13] Desbrun M, Tsingos N, Gascuel M-P. Adaptive sampling of implicit surfaces for interactive modelling and animation. Comput Graphics Forum 1996;15(5):319–25.
[14] Du H, Qin H. Direct manipulation and interactive sculpting of PDE surfaces. Comput Graphics Forum 2000;19(3):C261–70.
[15] Du H, Qin H. Dynamic PDE surfaces with flexible and general constraints. In Pacific Graphics 2001, Hong Kong; 2000. p. 213–22.
[16] Du H, Qin H. Integrating physics-based modeling with PDE solids for geometric design. In Pacific graphics, Tokyo, Japan; 2001. p. 198–207.
[17] Ebert DS, Musgrave FK, Prusinkiewicz P, Stam J, Tessendorf J. Simulating nature: from theory to practice. SIGGRAPH 2000 Course notes 25; 2000.
[18] Ferley E, Cani M, Gascuel J. Practical volumetric sculpting. Visual Comput 2000;16(8):469–80.
[19] Foster N, Metaxas D. Realistic animation of liquids. In Proceedings of GI; 1996. p. 204–12.
[20] Foster N, Metaxas D. Modeling the motion of hot, turbulent gas. In SIGGRAPH 1997, Los Angeles, CA, USA; 1997. p. 181–8.
[21] Frisken S, Perry R, Rockwood A, Jones T. Adaptive sampled distance fields: a general representation of shape for computer graphics. In SIGGRAPH 2000, New Orleans, USA; 2000. p. 249–54.
[22] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. In SIGGRAPH 1992; 1992. p. 71–8.
[23] Hua J, Qin H. Haptic sculpting of volumetric implicit functions. In Pacific Graphics 2001, Tokyo, Japan; 2001. p. 254–64.
[24] Hua J, Qin H. Dynamic implicit solids with constraints for haptic sculpting. In Shape Modeling International 2001, Banff, Alberta, Canada; 2002. p. 119–28.
[25] Lorensen W, Cline H. Marching cubes: a high resolution 3D surface construction algorithm. In SIGGRAPH 1987; 1987. p. 163–9.
[26] Morse B, Yoo T, Rheingans P, Chen D, Subramanian K. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In Shape Modeling International 2001, Genova, Italy; 2001. p. 89–98.
[27] Muraki S. Volumetric shape description of range data using blobby model. Comput Graphics 1991;25(4):227–35.
[28] Museth K, Breen D, Whitaker R, Barr A. Level set surface editing operators. In SIGGRAPH 2002, San Antonio, TX, USA; 2002. p. 330–8.
[29] Ohtake Y, Belyaev A, Alexa M, Turk G, Seidel H-S. Multi-level partition of unity implicits. In SIGGRAPH 2003, San Diego, USA; 2003. p. 463–70.
[30] Perry R, Frisken S. Kizamu: a system for sculpting digital characters. In SIGGRAPH 2001, Los Angeles, USA; 2001. p. 47–56.
[31] Press WH, Teulolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C. Cambridge University Press; 1993.
[32] Raviv A, Elber G. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In Proceedings of 5th ACM Symposium on Solid Modeling and Applications, Ann Arbor, Michigan, United States; 1999. p. 246–257.

[33] Sarti A, Tubaro S. Multiresolution implicit object modeling. In VMV 2001, Stuttgart, Germany; 2001. p. 93–100.
[34] Savchenko VV, Pasko AA, Okunev OG, Kunii TL. Function representation of solids reconstructed from scattered surface points and contours. Comput Graphics Forum 1995;14(4):181–8.
[35] Schneider R, Kobbelt L. Generating fair meshes with G1 boundary conditions. In Geometric Modeling and Processing Conference Proceedings; 2000. p. 251–61.
[36] Sclaroff S, Pentland A. Generalized implicit functions for computer graphics. Comput Graphics 1991;25(4):247–50.
[37] Stam J. Stable fluids. In SIGGRAPH 1999, Los Angeles, CA, USA; 1999. p. 121–7.
[38] Strang G. Introduction to applied mathematics. Wellesley-Cambridge Press; 1986.
[39] Turk G, Dinh HQ, O'Brien J. Implicit surfaces that interpolate. In Shape Modeling International 2001, Genova, Italy; 2001. p. 62–71.
[40] Turk G, O'Brien J. Shape transformation using variational implicit functions. In SIGGRAPH 1999, Los Angeles, CA, USA; 1999. p. 335–42.
[41] Turk G, O'Brien J. Modeling with implicit surfaces that interpolate. ACM Transact Graphics 2002;21(4):855–73.
[42] Ugail H, Bloor MIG, Wilson MJ. Techniques for interactive design using the PDE method. ACM Transact Graphics 1999;18(2): 195–212.
[43] Whitaker R, Breen D. Level-set models for the deformation of solid objects. In Conference of Implicit Surface 1998, Seattle, USA; 1998. p. 19–36.
[44] Witkin A, Heckbert P. Using particles to sample and control implicit surfaces. In SIGGRAPH 1994; 1994. p. 269–77.
[45] Zhang JJ, You L. Surface representation using second, fourth and mixed order partial differential equations. In Shape Modeling International 2001, Genova, Italy; 2001. p. 250–6.
[46] Zhao HK, Osher S, Fedkiw R. Fast surface reconstruction using level set method. In IEEE Workshop on Variational and Level Set Methods (VLSM 01) Vancover, Canada; 2001. p. 194–202.
[47] Zhao HK, Osher S, Merriman B, Kang M. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. Comput Vision Image Understanding 2000;80(3):295–319.

**Haixia Du** is a PhD candidate in the Computer Science Department at the State University of New York (SUNY) at Stony Brook, where she is also a Research Assistant in the Center for Visual Computing (CVC), SUNY at Stony Brook. Haixia received her BS degree in Computer Science from Jilin University in Changchun, P.R. China in 1995 and her ME degree in Computer Science from Institute of Mathematics, Chinese Academy of Sciences in Beijing, P.R. China in 1998. She also received her MS degree in Computer Science from SUNY at Stony Brook in 2000. Her research interests include geometric and physics-based modeling, computer animation and simulation, visualization, and computer graphics. For more information, see http://www.cs.sunysb.edu/~dhaixia

**Dr Hong Qin** is an Associate Professor of Computer Science at the State University of New York at Stony Brook, where he is also a member of the SUNYSB Center for Visual Computing. He received his BS degree (1986) and his MS degree (1989) in Computer Science from Peking University in Beijing, P.R. China. He received his PhD degree (1995) in Computer Science from the University of Toronto. From 1989–1990 he was a research scientist at North-China Institute of Computing Technologies. From 1990–1991 he was a PhD candidate in Computer Science at the University of North Carolina at Chapel Hill. From 1996–1997, he was an Assistant Professor of Computer and Information Science and Engineering at the University of Florida. In 1997, Dr Qin was awarded the NSF CAREER Award from the National Science Foundation (NSF), and, in September, 2000, was awarded a newly-established NSF Information Technology Research (ITR) grant. In December, 2000, he received a Honda Initiation Grant Award, and, in April, 2001, was selected as an Alfred P. Sloan Research Fellow by the Sloan Foundation. He is a member of ACM, IEEE, SIAM, and Eurographics.