Point Set Surface Editing Techniques based on Level-Sets

Xiaohu Guo Jing Hua Hong Qin
Department of Computer Science
State University of New York at Stony Brook
Email: {xguo|jinghua|qin@cs.sunysb.edu}

Abstract

In this paper we articulate a new modeling paradigm for both local and global editing on complicated point set surfaces of arbitrary topology. In essence, the proposed technique leads to a novel point-set methodology that can unify the topological advantage of the level-set methods and the simplicity of point-sampled surfaces. Any user-specified region of a point set surface in our system can be embedded into a grid-based level-set framework. The super-imposed grid structure enables both powerful local surface editing and global scalar-field free-form deformation anywhere across the point-sampled geometry. Furthermore, the underlying level-set representation, coupled with the concept of digital topology, greatly facilitates the topological modification of the sculpted point-set geometry whenever necessary during shape deformation. We have developed a variety of editing toolkits that can allow users to directly manipulate the point-set surface through interactive sketching, smoothing, embossing, and global free-form deformations with ease. We demonstrate the usefulness and efficacy of our prototype system for the point-sampled geometry via many examples.

1 Introduction

Most recently, point set surfaces are gaining momentum and enjoying their renaissance in both modeling and rendering. In the recent past, many efforts have been mainly concentrated on both direct rendering techniques [17], [24] and effective modeling mechanisms [1], [23], [15], [16] for point sampled geometry without worrying about their connectivity. However, such research achievements are still not fully adequate enough for interactive shape control of the point based geometry, and the spectrum of surface editing and deformation types is still quite limited. This paper aims to serve this need by developing the level-set-based editing paradigm for point set surfaces in order to enable both local manipulation and global free-form deformation over any region of interest across the entire point-set surface.

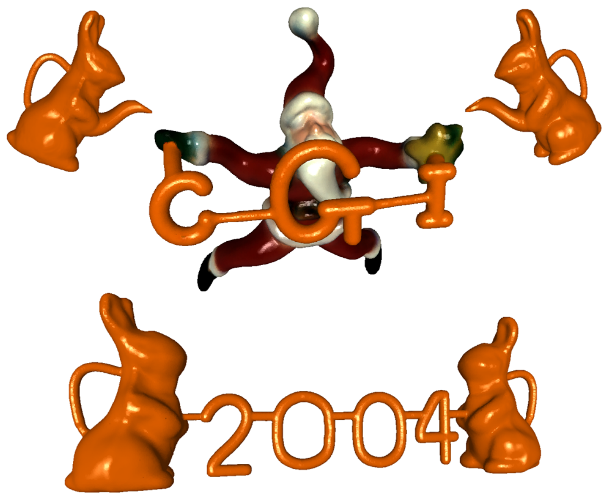
Figure 1. “Rabbit Teapots” and “CGI 2004” logo created using our level-set-based point set surface editing framework.

Our novel editing techniques are founded upon the representation of an embedding scalar field associated with any region of the point-set surface. In comparison with the pure, explicit point-based surface representation, the scalar-field-driven implicit representation and level sets have been proven as a very powerful paradigm that can not only be free of parameterization artifacts, but also handle arbitrary topology and complicated geometry easily. In particular, the implicit representation further facilitate the proper topological change during the shape sculpting process without any ambiguity. We shall demonstrate the ease of the topological modification for the modeled objects later in our paper. In our level-set based surface editing framework, we employ unstructured point samples as the basic modeling and rendering primitive throughout this paper. By constructing an implicit surface from the point cloud using some existing techniques [13], we can naturally incorporate the level-set-based surface editing techniques [12] into the deformation framework of the point based geometry, which not only offers us a wide range of powerful surface editing techniques for the point set surface editing, but also facilitates the topology change with ease. Equipped with the level-set editing functionality, Scalar Free-Form Deformations (SFFD) [10] can be also incorporated into our surface deformation system in order to further expand the ability for both local and global surface editing. With the aid of dynamic re-sampling, we can quickly update the surface shape of the point-based geometry (undergoing geometric deformation and/or topological change) without worrying about point connectivity at all.

2 Previous work

This section briefly outlines the background of point set geometry processing, level-set methods, and free-form deformations. In the interest of space limitation, a more comprehensive survey on these topics is not feasible for this conference paper.

In recent years, considerable research has been devoted to the efficient modeling, and processing of point-sampled geometry. In [23], the Pointshop 3D system was presented for interactive shape and appearance editing of 3D point sampled geometry. Alexa et al. [1] used the framework of moving least squares (MLS) projection to approximate a smooth surface defined by a set of points. Pauly et al. [15] presented several efficient simplification schemes for the point sampled surfaces. Later, they presented a free-form shape modeling framework [16] for point sampled geometry using the implicit surface definition of the moving least squares approximation. Most recently, Guo and Qin [9] presented a physics-based dynamic local sculpting paradigm for point sampled surfaces using volumetric implicit functions.

Level-set methods were introduced by Osher and Sethian [14] by representing the contour as the level-set of a scalar-valued function. Desbrun et al. [8] and Breen et al. [6] used this method for shape morphing. Whitaker [21] and Zhao et al. [22] employed this method for 3D reconstruction. More recently, Museth et al. [12] and Baerentzen et al. [2] presented the level-set frameworks for interactively editing implicit surfaces.

Free-form shape deformations have been studied extensively in the past [3], [18], [7], [19]. Recently, Hua and Qin [10] proposed the Scalar-field Free-Form Deformation (SFFD) technique based on general flow constraints and implicit functions. In this paper we further extend the SFFD approach and integrate it into our level-set framework in order to provide users the powerful free-form deformation tools directly on point-set surfaces.

3 System overview

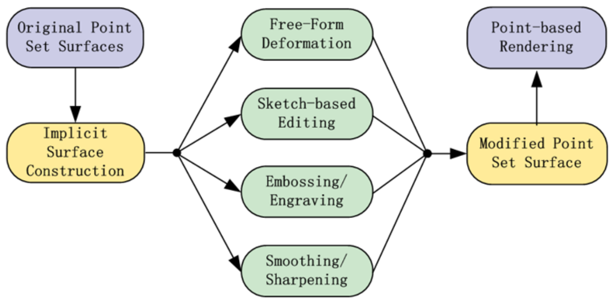

Figure 2. The editing framework.

The data flow and architecture of our level-set based point set surface editing system is presented in Figure 2. The system takes any region of the original point set surfaces as the input, utilizes the level-set based editing techniques to deform the point set either locally or globally and change its topology when needed, and finally uses point-based surface rendering techniques to render the modified surface. In this paper, we utilize the Multi-level Partition of Unity (MPU) method [13] to convert any user-specified region of the original point set surfaces into an implicit surface representation. It may be noted that even after the implicitization process, the original point-sampled geometry must be retained for the downstream procedures such as local editing, global free-form deformation, topological change, and rendering. After the construction of the implicit surface, the existing level-set surface editing techniques [12] can be directly integrated into our system, such as sketch-based editing, embossing/engraving, smoothing/sharpening, etc. While incorporating the level-set operators, we also integrate the Scalar-field Free-Form Deformation (SFFD) techniques [10] into the level-set framework to expand the ability for both local and global surface editing. After the deformation of the underlying level-set surface at each iteration step, we update the point set surface accordingly. Besides the usual geometric deformation, collision detection and topology change should be considered, which we address in Section 4.4.

4 Level-set-driven surface modeling

4.1 The level-set formulation

Deformable iso-surfaces implemented through the mean of level-set methods [14] have demonstrated a great potential in a wide range of fields such as visualization, scientific computing, computer graphics, and vision related areas. An implicit surface consists of all points

$$S = \{\vec{x}(t) | \phi(\vec{x}, t) = k\},$$

where $\phi(\vec{x}, t)$ is a time-varying scalar function embedded in 3D. Level-set methods relate the motion of the implicit surface to a partial differential equation (PDE) defined on the associated volume through:

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| F(\vec{x}, \phi, \nabla \phi, ...), \tag{1}$$

where $\nabla \phi$ denotes the gradient of the implicit surface and $F(\vec{x}, \phi, \nabla \phi, ...) \equiv -\frac{\nabla \phi}{|\nabla \phi|} \cdot \frac{d\vec{x}}{dt}$ is the speed function. Using the level-set method, we can implement a wide range of deformations by defining an appropriate speed function, which is always based on a combination of data dependent terms, and geometric measures (e.g. curvature) of the implicit level-set surface [20][11]. In this paper, we not only utilize this representation to perform the level-set-based editing operations on the point set surfaces, but also integrate the Scalar-Field Free-Form Deformation [10] into our level-set framework.

Since we are only interested in the surface editing of the point set which are relying on the zero level-set of the iso-surfaces, first we only need to solve the parts of the solution that are adjacent to the moving surface. Second, our operations are directly applied to the point set, so the level-set embedding occurs wherever the deformation is needed. The level-set space and the deformation operators within the space are controlled by users interactively. In this paper, we utilize the Sparse-Field method proposed by Whitaker [21] to update only the wavefront, and several layers around it via a simple city block distance metric at each iteration. The set of grid points adjacent to the level-set is called *active set* denoted by L_0, and the neighborhoods of the active set are defined in layers, $L_{+1}, ..., L_{+l}, ..., L_{+N}$ and $L_{-1}, ..., L_{-l}, ..., L_{-N}$, where the l indicates the city block distance from the nearest active grid point(Note that, negative numbers are used for the outside layers). The work in this paper uses only up to the second-order derivatives of ϕ, so we only need 5 layers L_2, L_1, L_0, L_{-1}, and L_{-2}. For more detailed information on the Sparse-Field method, please refer to [21]. The complexity of the computational time grows only proportional to the area of the surface region which undergoes the deformation.

4.2 Free-form deformation

Free-Form Deformation has been used extensively in shape modeling and animation since the designers do not need to worry about the underlying geometric representation and topological structure of the embedded object. It would be rather straightforward for users to perform the Free-Form Deformation directly on the point set surface as in [16]. However, then we would have to rely on the Moving Least Squares surface projection to find out the exact position for all the newly inserted points. This is because point insertion is unavoidable if a deformation is large and in fact expands the model rather significantly. As a result, gaps will occur at the current resolution. In principle, point insertion is far from trivial. To address this problem, we take a rather different approach by performing the Free-Form Deformation directly on the global scalar field as we acquired using the Multi-level Partition of Unity (MPU) implicit surface construction methods[13]. Then the global scalar field can be used to guide both the point movement and point insertion in a single unified fashion, without the separate effort for adding new points.

Specifically in this paper, we utilize the Scalar-field Free-Form Deformation (SFFD) introduced in [10]. Since it is founded upon the PDE-based flow constraints, we can easily convert the velocity field obtained by SFFD into the updating of the global scalar field of our point surface. Furthermore, introducing intermediate steps during deformation steps allows us to perform dynamic sampling on the point set surface in order to maintain a good surface quality throughout the model sculpting session. So in our representation, we have two classes of scalar field. One is for the surface representation, as we denoted by s_s, which is exactly the global scalar field we acquired from the point clouds using the MPU implicit surface construction method. Another scalar field is for the deformation tool, as we denoted by s_t, which is the scalar field of the deformation tool described in [10], such as the sketched point or curve skeleton.

4.2.1 Scalar-field free-form deformation

Now we shall briefly overview the idea of applying the scalar field s_t of the tools to perform deformations on the scalar field s_s of the point surface. In our global free-form deformation, we are still using the discretized voxel grids to store both s_s and s_t. And we utilize the sparse-field method to update them at each iteration step.

During the deformation of the tool scalar field s_t, we assume that the vertices on the discretized voxel grids are constrained on the same level-set where they originally reside. Then the trajectory of these grid vertices can be represented as:

$$\{\vec{x}(t) | s_t(\vec{x}(t), t) = c\}.$$

The derivative of $s_t(\vec{x}(t), t)$ with respect to time yields:

$$\nabla_{\vec{x}} s_t \cdot \frac{d\vec{x}}{dt} + \frac{\partial s_t}{\partial t} = 0,$$

where $\nabla_{\vec{x}} s_t$ is the gradient of s_t at $\vec{x}$. In order to get the general velocity along the three coordinate axes of the 3D space (v_x, v_y, v_z), and also take the smoothness of the deformation motion into account, we can add a smoothness

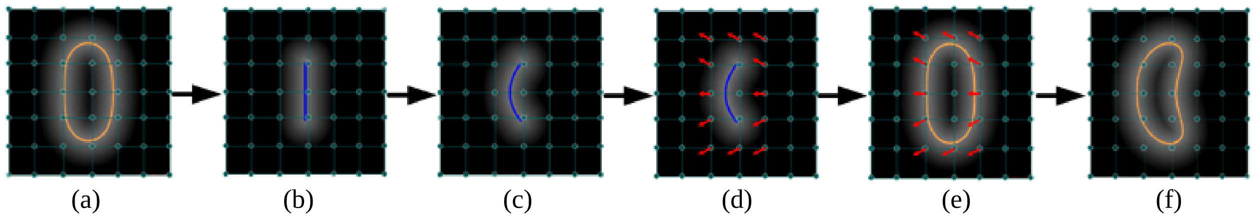

Figure 3. (a) The surface scalar field s_s representing the point set surface; (b) The initial tool scalar field s_t; (c) The deformed tool scalar field; (d) The velocity (red arrow) field computed according to the deformation of the tool scalar field; (e) The velocity field is utilized to deform the surface scalar field; (f) The deformed surface scalar field.

constraint on the underlying level-set surface s_s by minimizing the following objective function:

$$E = \int (\nabla_{\vec{x}} s_t \cdot \vec{v} + \frac{\partial s_t}{\partial t})^2 + \lambda(|\nabla \vec{v}|)^2 d\vec{x}, \quad (2)$$

where λ is a Lagrange multiplier. In order to discretize the objective function 2, we can consider a voxel grid vertex i, where $i \in L_{-2} \cup L_{-1} \cup L_0 \cup L_1 \cup L_2$ (The Sparse-Field Layers), and its adjacent neighboring voxel grids N_i, where $N_i = \{j | \overline{ij} \in C_6, j \in L_{-2} \cup L_{-1} \cup L_0 \cup L_1 \cup L_2\}$, C_6 is the set of 6-connected voxel grid pairs. Then we can transform the objective function 2 into:

$$E = \sum_i (c(i) + \lambda s(i)), \quad (3)$$

where $c(i)$ is the error of the flow constraint approximation:

$$c(i) = (\frac{\partial s_t}{\partial x} v_x(i) + \frac{\partial s_t}{\partial y} v_y(i) + \frac{\partial s_t}{\partial z} v_z(i) + \frac{\partial s_t}{\partial t})^2,$$

and $s(i)$ is the discretized smoothness factor computed as the velocity difference between the voxel grid vertex and its adjacent neighbors:

$$s(i) = \frac{1}{|N_i|} \sum_{j \in N_i} [(v_x(i) - v_x(j))^2 + (v_y(i) - v_y(j))^2 + (v_z(i) - v_z(j))^2],$$

where $|N_i|$ is the number of voxel grid vertices in N_i. By satisfying $\frac{\partial E}{\partial v_x(i)} = 0$, $\frac{\partial E}{\partial v_y(i)} = 0$, and $\frac{\partial E}{\partial v_z(i)} = 0$, we can minimize the objective function E and obtain the iterative solution:

$$[v_x, v_y, v_z]^\top = [\overline{v}_x, \overline{v}_y, \overline{v}_z]^\top - \mu [\frac{\partial s_t}{\partial x}, \frac{\partial s_t}{\partial y}, \frac{\partial s_t}{\partial z}]^\top, \quad (4)$$

where $(\overline{v}_x, \overline{v}_y, \overline{v}_z)$ is the average velocity of all the adjacent voxel grids, and

$$\mu = \frac{\frac{\partial s_t}{\partial x}\overline{v}_x + \frac{\partial s_t}{\partial y}\overline{v}_y + \frac{\partial s_t}{\partial z}\overline{v}_z + \frac{\partial s_t}{\partial t}}{\lambda + (\frac{\partial s_t}{\partial x})^2 + (\frac{\partial s_t}{\partial y})^2 + (\frac{\partial s_t}{\partial z})^2}.$$

4.2.2 Updating surface scalar field and point geometry

It may be noted that, although we obtain the velocity field associated with each voxel grid according to the change of the tool scalar field s_t, we are not going to change their positions. Instead, we utilize the velocity field $\vec{v}$ acquired above to update the surface scalar field s_s by defining the speed function in the following level-set fashion:

$$F = -\frac{\nabla s_s}{|\nabla s_s|} \cdot \vec{v}. \quad (5)$$

Based on these formulations, we can design our level-set based global free-form deformation as the following evolution process. First we generate a tool scalar field using skeletons or sketches. Also, we track down the original tool scalar values s_{to} at these voxel grid positions. After the user alters the tool scalar field, we also track down their final tool scalar values s_{tf}. Then $s_{tf} = s_{to} + k\Delta s_t$, while k is a user specified number of steps taken to deform s_t, and Δs_t is the step size of the tool scalar field evolution.

The SFFD algorithm for point geometry is as follows:

At each time step m (m=0...k):

1. Update the tool scalar field by: $s_t = s_{to} + m\Delta s_t$, then $\frac{\partial s_t}{\partial t} = \Delta s_t$;
2. Calculate $\frac{\partial s_t}{\partial \vec{x}}$ with finite difference;
3. Initiate the velocities of the voxel grids in the Sparse-Field layers to be 0;
4. Deduce the current velocity field by iteration using Equation (4). This normally needs 3-4 iterations for the velocities to converge;
5. Deduce the updating rate of the surface scalar field by Equation (5), and obtain the updated surface scalar field s_s;
6. Update points' positions and perform dynamic sampling (see Section 4.3 for details).

7. If m=k, terminate the deformation process; Otherwise, proceed to the next time step $m + 1$ and repeat the above steps.

Figure 3 illustrates the idea of deforming the surface scalar field based on the deformation of tool scalar field. In this figure, the yellow curves denote the iso-surface of the surface scalar field associated with the point set surface, and the blue curves denote the skeleton-based tools used to define the tool scalar field. The red arrows in (d) and (e) denote the velocities evaluated at those voxel grids of the Sparse-Field Layers.

4.3 Dynamic update of point set surface

After we modify the surface scalar field through user interaction, we must change the points' locations since the point samples are assumed to be on the zero level-set of the underlying implicit function. As in [9], if we assume that the points are only moving in their normal direction, we can obtain its normal velocity:

$$\vec{v}_{\vec{n}} = \frac{\frac{\partial s_s}{\partial t}}{|\nabla_{\vec{x}} s_s|}\vec{n}. \tag{6}$$

where $\nabla_{\vec{x}} s_s$ denotes the gradient $\frac{\partial s_s(\vec{x}(t),t)}{\partial \vec{x}}$, and $\vec{n}$ is the normal of the surface evaluated at $\vec{x}$. This velocity is then employed to update the point position by advancing to the next time step through the forward Euler method:

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_{\vec{n}} \Delta t.$$

The point sampling density will be changed while users are performing sculpting or deformation on the surface. To maintain a nice surface quality, we need to insert new sample points while the surface density becomes too low, or we can simplify the surface by eliminating points while the surface is squeezed otherwise. We use the up-sampling scheme from [1] for the point insertion. In each modeling step, each point should check its neighboring density by projecting its neighbor points onto its tangent plane. Then we compute the Voronoi diagram of these points. We choose the Voronoi vertex that has the largest circle radius on the tangent plane. If the radius is larger than a specified threshold, we can project the vertex onto the iso-surface of the global scalar field. Using this approach, we can achieve a surface density of locally near-uniform. In the meantime, we can also reduce the sampling density using any method proposed in [15]. We implement the iterative simplification method similar to the one introduced in [15]. Instead of using a quadric error metric, we simply replace two points by their middle point and then project this new point onto the iso-surface of the global scalar field. The up-sampling and down-sampling process can be followed by a relaxation stage using a simple point repulsion scheme similar to [16] to obtain a more uniform sampling pattern.

4.4 Topology change handling

In order to use our level-set based surface editing tools to change the shape and topology of the point set surfaces, the surface must be able to change its topology properly whenever a collision with other parts of itself is detected. The underlying level-set surface contour can automatically change topology (e.g. merge or split) without requiring an elaborate mechanism to handle such changes. However, the topology of the point set surfaces need to be explicitly handled, e.g. deleting the points in the intersection region. So we need a robust method to predict where the topology change is going to be occurring.

In [5], an efficient and robust method to control the topology of the level-set contour was proposed by adding topological information to the volume representation with a clear goal of locally resolving topological ambiguities. Their novel method is extremely useful for collision detection of the level-set contour. In our current work, our tasks are different and our focus is mainly on the robust and proper topological modification of the point-sampled geometry undergoing deformation. Therefore, we simply change the topology when a collision is detected, even though it is quite straightforward to integrate the whole settings of [5] into our framework to preserve the topology of the underlying point set surface. In the interest of space limitation, more details about the digital topology can be found in [5] or elsewhere.

According to [5], we say that a grid point $\vec{v}$ is simple with respect to $V \subset Z^3$, if V and $V \cup \{\vec{v}\}$ have the same number of components, handles and cavities. Otherwise $\vec{v}$ is called complex. Furthermore, [4] proved that $\vec{v}$ is simple with respect to V if and only if

$$n_{int}(\vec{v}, V) = n_{ext}(\vec{v}, V) = 1. \tag{7}$$

where $n_{int}(\vec{v}, V)$ is the number of interior components of V that touch $\vec{v}$, and $n_{ext}(\vec{v}, V)$ the number of exterior components. [5] gives more detailed definitions of $n_{int}(\vec{v}, V)$ and $n_{ext}(\vec{v}, V)$ based on digital topology, and more theoretical results are available in [4]. However, the above discussion is sufficient to enable the development of modeling and deformation tools to correctly handle collision detection and topological modification for the point-set surfaces.

For the purpose of clarity, we only consider the situation when the surface is advancing outward, since for the inward case, we can simply switch the role of the inside and outside of the surface region. A grid point is said conquered if it is either on the active layer L_0 of the sparse-field (the green circle grids in Figure 4) or inside the surface region (the blue square grids in Figure 4). Whenever a grid point is about to be conquered (i.e. changing its status from the outside layer L_{-1} to the active layer L_0, e.g. the red triangle grid in Figure 4), we first compute its n_{int} and n_{ext} using the digital topology method provided in [5], and test it for simplicity according to Equation (7). If it is simple, the Sparse-Field algorithm proceeds as usual. If not, we have to explicitly resolve the topology change of the point set surface, i.e. deleting the point samples residing in the small vicinity of this grid. Similar to [5], the only case we need to handle the topological change and resolve the topological ambiguity is when $n_{int} \geq 2$, i.e. two parts of the front collide. Figure 4 shows an example of the collision between two parts of the front(green solid curves). The red dashed curve will become the new advancing front in the current iteration step, while grid $\vec{v}$ is changing its status from L_{-1} to L_0. The topology number n_{int} for $\vec{v}$ equals 2. So we shall delete the point samples residing in the prescribed vicinity of grid $\vec{u}$ and $\vec{w}$ (Note that, the affected region can be identified either interactively or automatically given the user-specified parameters such as the influence factor of each grid). Also we need to place a tag between $\vec{v}$ and $\vec{w}$, so they will be considered disconnected when checking the topology numbers for other grid points nearby.

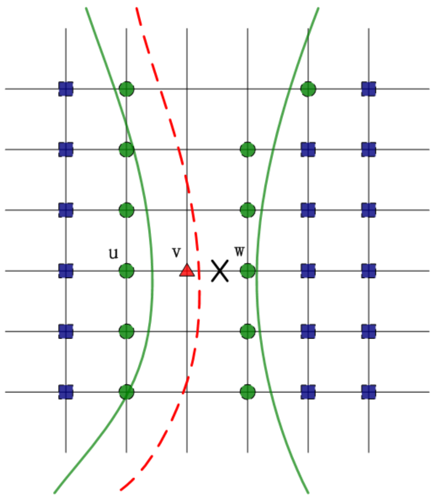

Figure 4. Collision detection.

5 Editing toolkits

5.1 Free-form deformation tools

In order to enable users to perform free-form deformations on the point set surfaces, our system allows users to interactively sketch skeletons using a mouse or a 3D pointing device. Then the tool scalar field is generated as the blending of field functions g_i of a set of skeletons $t_i (i = 1, ..., N)$:

$$s_t(x, y, z) = \sum_{i=1}^{N} g_i(x, y, z),$$

where the skeletons t_i can be any geometric primitive such as blobs, curves, etc. After the construction of the tool scalar field s_t, the designers can enforce global control of s_t in two different ways: (1) adjusting the coefficients of the implicit functions defined for each skeletal element; (2) manipulating or moving the skeletons. When the designers modify the tool scalar field, the embedded surface scalar field and the point set surfaces are deformed according to our aforementioned SFFD algorithm. Figure 5(a) shows the example of performing free-form deformations on a rabbit model by adjusting the influence radius of the underlying blob skeletons, where the blue color indicates the isosurface of the skeleton-based scalar field. Figure 5(b) shows the example of bending the rabbit model by bending the underlying curve skeleton. Note that bending the underlying curve skeleton first changes the tool scalar field, then it leads to deformation of the embedded point geometry according to the algorithm documented in Section 4.2.2. Users can also perform tapering operations by simply sketching source and target strokes, see Figure 5(c).

5.2 Sketch-based editing tools

We also develop several simple sketching tools to allow users to directly work on point set surfaces with hand strokes. Strokes can be gathered from the mouse as a set of curves or a collection of points and the Gaussian blobs are assigned evenly along the curve or at each point. The surface will then grow along this implicitly defined region, which can be either growing outside or drilling inside depending on the direction of the surface motion. Here we utilize the commonly used speed function which consists of a combination of two terms [20] [11]:

$$F = \alpha D + (1 - \alpha)\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|},$$

where D is a term dependent on the user input strokes, and it forces the surface to expand or contract toward the boundary of the stroke region. The term $\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$ is the mean curvature H of the surface, which forces the surface to remain smooth. The topology change handling techniques addressed in Section 4.4 is employed to handle collision detection and point sample deletion. Figure 6 shows an example of the sketch-based surface growing method. By inverting the growing direction of level-set surfaces, we can easily achieve drilling operations based on the user's input strokes.

5.3 Other tools

The embossing/engraving and smoothing/sharpening operations can be easily achieved by utilizing the speed function proposed by [12]:

$$F = D_q(d)C(\gamma)G(\gamma).$$

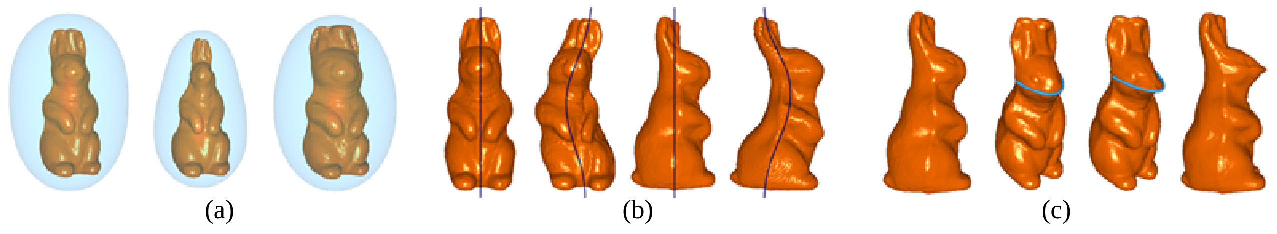

Figure 5. (a)Shrinking and Inflation of a rabbit model using blobs skeleton; (b)Bending of a rabbit model using curve skeleton; (c)Tapering of the rabbit mouth using sketched strokes.

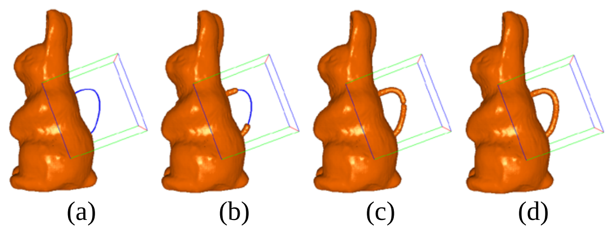

Figure 6. Sketch-based surface growing.

Here D_q is a distance-based cut-off function that depends on a distance measure d to a geometric region of influence (ROI) primitive, e.g. a superellipsoid. $C(\gamma)$ is a cut-off function that controls the contribution of $G(\gamma)$ to the speed function, while $G(\gamma)$ is dependent on geometric measures γ of the level-set surface, e.g. curvature.

For embossing/engraving operations, the user can draw curves, or place some points near the surface, which are used to attract/repel the surface. The surface can be also smoothed/sharpened by applying motions in a direction that reduces/increases the local surface curvature. For more detailed information on the formulation of the speed functions for embossing/engraving and smoothing/sharpening, please refer to [12]. Figure 7 (a) shows an example of surface embossing based on a set of user sketched curves near the surface. Figure 7 (b) demonstrates the smoothing operator applied to the intersection region of the point set surfaces constructed from the CSG union operations.

6 Results and Time Performance

Both simulation and rendering parts of our system are implemented on a Microsoft Windows XP PC with dual Intel Xeon 2.0GHz CPUs and 1.5GB RAM. We document the various point set surface editing techniques in our system and report their time performances in Table 1. We did not include the dynamic sampling time into the updating time in Table 1, because the dynamic sampling time greatly depends on the number of points inserted/deleted on each simulation step. In our experiment of global FFD of the rabbit model (67,038 points), the required time of checking all the point samples for dynamic sampling is around 0.5 second.

Using our level-set-based point set surface editing framework, we have created several interesting objects. See Figure 1. The "rabbit teapots" are created from rabbit models and spouts of the teapot models by using curve-based bending on the rabbit models, tapering on the rabbits' mouths, sketch-based editing for the handles on the back of the rabbits, and smoothing the intersection of the rabbits and the spouts after the CSG union operation between them. The "CGI 2004" logos are created using sketch-based editing techniques by sketching the logo-shaped curves on the santa and rabbit point set surface models and allowing the surface to grow along the user input curves.

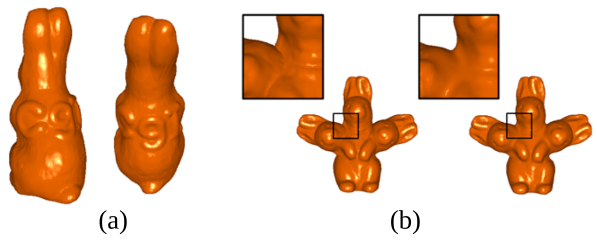

Figure 7. (a)Surface embossing based on user sketched curves; (b)Curvature-based surface smoothing.

Table 1. The simulation time of our point set surface editing tools applied on the rabbit model (67,038 points).

Editing Tools	# Grids	# Points	Time (s)
Shrinking/Inflation	256^3	67,038	7.809316
Bending	256^3	67,038	7.631438
Tapering	128^3	24,107	1.815174
Sketch Editing	128^3	5,188	0.323902
Embossing	128^3	5,824	0.370414
Smoothing	128^3	7,189	1.647415

7 Conclusion

In this paper, we have developed a new modeling and deformation paradigm for point set surfaces and articulated its key constituents and the associated contributions. Our novel paradigm facilitates both local and global editing on arbitrarily complicated point set surfaces of any topological types. We have implemented a unique modeling framework that collectively takes advantage of level-set geometry and scalar-field-driven free-form deformation. The key feature of our framework is a new point-geometry methodology that uniquely integrate the topological flexibility of the level-set approach and the simplicity of point-sampled surfaces. The grid structure resulted from the level-set approach enables both powerful local surface editing and global scalar-field free-form deformation anywhere across the point-sampled geometry. Furthermore, we employ the techniques of digital topology to handle topological changes during the shape deformation. We have developed a family of editing toolkits such as interactive sketching, smoothing, embossing, and sharpening. It is our hope that, through our extensive experimental results, we can show that our new approach on point set surfaces is both promising and valuable for interactive graphics.

Several further improvements to extend our current research work are possible in the near future. Currently the speed of our editing system is limited by the dynamic sampling rate and the spatial resolution of the underlying level-set approach. More efficient algorithms with the improved performance are always desirable. Our ultimate goal is to enhance all the deformation operations with haptics, including the free-form deformation and the local shape editing, so that the users can obtain the realistic force feedback when performing all of the previously-mentioned deformations on any point set surfaces.

Acknowledgment

This research was supported in part by the NSF grants IIS-0082035 and IIS-0097646, and Alfred P. Sloan Fellowship. The santa and rabbit models are courtesy of Cyberware Inc.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE TVCG*, 9(1):3–15, January-March 2003.

[2] J. A. Baerentzen and N. J. Christensen. Volume sculpting using the level-set method. In *International Conference on Shape Modeling and Applications*, pages 175–182, 2002.

[3] A. H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH*, pages 21–30, 1984.

[4] G. Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. In *Pattern Recognition Letters*, pages 1003–1011, 1994.

[5] S. Bischoff and L. Kobbelt. Sub-voxel topology control for level set surfaces. In *Eurographics*, pages 273–280, 2003.

[6] D. E. Breen and R. T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Trans. on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[7] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *SIGGRAPH*, pages 187–196, 1990.

[8] M. Desbrun and M.-P. Cani. Active implicit surface for animation. In *Graphics Interface*, pages 143–150, Jun 1998.

[9] X. Guo and H. Qin. Dynamic sculpting and deformation of point set surfaces. In *Pacific Graphics*, pages 123–130, 2003.

[10] J. Hua and H. Qin. Free-form deformations via sketching and manipulating scalar fields. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, 2003.

[11] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. In *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, pages 158–175, 1995.

[12] K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr. Level set surface editing operators. In *SIGGRAPH '02 Proceedings*, pages 330–338, 2002.

[13] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH*, pages 463–470, 2003.

[14] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, November 1988.

[15] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. *IEEE Visualization*, 2002.

[16] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *SIGGRAPH*, 2003.

[17] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. *SIGGRAPH*, pages 343–352, 2000.

[18] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH*, pages 151–160, 1986.

[19] K. Singh and E. Fiume. Wires: A geometric deformation technique. In *SIGGRAPH*, pages 405–414, 1998.

[20] R. T. Whitaker. Volumetric deformable models: Active blobs. In *Visualization in Biomedical Computing*, pages 122–134, 1994.

[21] R. T. Whitaker. A level-set approach to 3d reconstruction from range data. In *International Journal of Computer Vision*, pages 203–231, 1998.

[22] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proc. 1st IEEE Workshop on Variational and Level Set Methods*, pages 194–202, 2001.

[23] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop3d: An interactive system for point-based surface editing. *SIGGRAPH*, 2002.

[24] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. *SIGGRAPH*, pages 371–378, 2001.