Vision, Modeling, and Visualization (2010)

Continuous Deformations of Implicit Surfaces

J. Martinez Esturo, C. Rössl, and H. Theisel

Visual Computing Group, Otto von Guericke University Magdeburg, Germany

Abstract

We introduce an approach for the continuous deformation of implicit surfaces which considers properties of all isosurfaces of a volume data set simultaneously. This is achieved by integrating divergence-free vector fields which is carried out by an efficient backward Lagrangian integration scheme. Our deformation guarantees volume preservation inside each isosurface as well as the preservation of continuity and topology of every isosurface. For visualization and interaction, we offer a real-time mode that allows interactive working on the resolution of the underlying volumetric grid as well as a grid resolution independent mode offering exact extraction of arbitrary isosurfaces. We apply the approach to the deformation of measured volume data sets as well as to the design of complex implicit shapes with a simple pre-defined topology.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Shape deformations are a standard problem in computer graphics. Shapes are usually given as surfaces, either in an explicit (e.g., as triangular mesh or as parametric surface) or an implicit form. Most of the existing approaches consider deformations of explicit surfaces. However, there are also a number of approaches for implicit surfaces which mostly focus on deforming one single isosurface.

Volume data, or more generally scalar fields, contain much more information than just one isosurface. In fact, there is a whole family of isosurfaces which may represent different kinds of information: in CT or MRI data from medical imaging applications, different isosurfaces describe transitions between different materials (like bone, tissue, or air), in other applications different isosurfaces may contain distance information to a particular isosurface of interest. Generally, for most volume data sets there is more than a single isosurface of interest.

If one is interested in the deformation of one particular isosurface, a generic solution is to extract it, then apply an explicit deformation, and finally perform an implicitation. Such an approach does not regard any volumetric information except for the location of one isosurface. However, if the complete volume is of interest, a good deformation should incorporate the whole field, i.e., it should take care of the shape of *all* isosurfaces. This paper presents a new approach to such deformations. The deformation is defined by a pathline integration of a constructed divergence-free vector field and has the following properties: the volume inside each isosurface is preserved during the deformation, no isosurface changes its topology, no new critical points of the volume data set appear or disappear, and a C^1 continuity of the isosurfaces is preserved during the deformation. The desired volume preservation is justified by the fact that many materials in nature preserve their volume under deformation. At the same time, topology preservation allows to construct complicated shapes with a pre-defined simple topology.

The deformation is computed numerically by an efficient backward Lagrangian integration scheme. For interactive real-time modeling we visualize isosurfaces on the grid of the underlying volume data set. In addition, an exact reconstruction can be used to retrieve an arbitrary isosurface with exact topology and high accuracy. We apply our technique to deform volume data sets and to model complex families of isosurfaces with a pre-defined simple topology.

The main contribution of the paper is the introduction of a volume deformation approach which preserves the volume, continuity and topology of every isosurface. To carry out the deformation, we introduce a backward Lagrangian integration scheme.

DOI: 10.2312/PE/VMV/VMV10/219-226

2. Related work

There are many approaches to shape modeling and deformation in computer graphics, and consequently there is a huge body of literature as well as a significant amount of ongoing research activity. In this section we briefly review related work focussing on volumetric modeling and implicit surface deformation as well as approaches which guarantee volume preservation.

The vast majority of deformation methods acts on *explicit* surfaces. Here, piecewise representations, especially triangle meshes (or more generally polygonal surfaces) and point-based models have emerged as de facto standard for shape models, and there exist a great variety of deformation methods. There are two general approaches to deforming explicit shapes: surface-based deformations and space deformations. Surface-based deformations act directly on the surface model. As we do not address surface-based deformations in this paper, we refer to the recent survey of Botsch and Sorkine [BS08] and the references therein. In contrast to surface-based deformations, space deformations pioneered by Sederberg and Parry [SP86] establish a mapping from the domain onto a warped space. Thus any shape embedded in the original domain can be mapped to a deformed version. For a detailed overview and further references we refer to [MJBF02] and the survey [GB08]. Furthermore, there are various hybrid methods (see, e.g., [Coh09]).

There are numerous tasks in computer graphics and geometry processing that are performed more easily on *implicit* surfaces rather than on explicit models: for instance, changes in surface topology are generally simpler, or self-intersections are avoided by construction. However, there is significantly fewer work on deformation of implicit surfaces than for the explicit case. Closely related are volume deformations, which are often modeled by a space deformation. For instance, in medical applications this includes non-rigid registration of data sets. Often the data describes certain material properties, like soft-tissue, and manipulations are required to be physically-based. We refer to the survey [CCI*07] for a general and broad review of volume deformations with a discussion on various data representations and applications.

Several classical approaches to modeling with isosurfaces are based on level set methods [OF01]. Museth et al. [MBWB02] define shape editing operations for smoothing, offsetting, and blending. Desbrun and Cani [DCG98] use a level set approach to define active implicit surfaces inspired by geometric snakes. Level set methods focused on explicitly handling topology preservation were presented in [AS05]. As these approaches act only on a particular isosurface, computations can be limited to a narrow band of the scalar fields. Various other approaches are physically plausible and employ particle systems and a Lagrangian integration scheme to simulate and animate surface material. In computer graphics this was pioneered by Desbrun and Cani-Gascuel [DG95] who minimize local volume variations. Another potential goal for such approaches is emulating virtual clay [MQW01,CA06]. Alternative sculpting methods implement virtual carving operations [PF01] based on adaptive distance fields [FPRJ00]. Yet a different approach to volume deformation consists in simulating networks of geometric primitives, e.g., using a chain mail analogy [Gib97] (efficient GPU-based variants are presented in [SBH07, RWE08]). A crucial aspect of any deformation method is interactivity: the user requires real-time feedback. The use of modern graphics hardware can speed up computations enormously or is even mandatory to achieve interactive frame rates on volume data [RSSG01, WR01, GW06], see also [HKRW06] for an overview.

Finally, we emphasize deformation methods which are designed to guarantee volume preservation globally. Contrary to our work, all these approaches act on explicit surfaces. Angelidis et al. [AWC04] define space deformations by sweeps of shapes which serve as tools. While the focus here lies on avoiding fold-overs, the same authors develop a variant based on volume preserving swirls [ACWK04], which also preserves global volume. Von Funck et al. [vFTS06, vFTS07] apply divergence-free vector fields which define the trajectories of vertices deforming an explicit surface. This approach is volume preserving by construction.

3. Approach

Given an initial 3D scalar field $s_0(\mathbf{x})$ over a spatial domain D, we consider a continuous deformation over time as finding a time-dependent scalar field $s(\mathbf{x},t)$ with $s(\mathbf{x},t_0) = s_0(\mathbf{x})$. We define the deformation by a 3D time-dependent vector field $\mathbf{v}(\mathbf{x},t)$ describing the transport of the isosurfaces over time. We use the concept of a *flow map* ϕ of $\mathbf{v}$ which is the map from the point where a massless particle is seeded at time t to the point where it is located at time τ under a pathline integration of $\mathbf{v}$:

$$\phi_t^\tau(\mathbf{x}) = \mathbf{x} + \int_t^\tau \mathbf{v}(\mathbf{x},u)\, du.$$

Given s_0 and $\mathbf{v}$, finding the deformation s means to solve the PDE

$$\frac{\partial s}{\partial t} = -(\nabla s)^T \mathbf{v} \quad , \quad s(\mathbf{x},t_0) = s_0(\mathbf{x}) \tag{1}$$

where ∇ describes the (spatial) gradient. Equation (1) is known as the *fundamental level set equation* [OF01], the solution is an initial value problem.

There are a variety of approaches to solve (1) which are based on a discretization of s and $\mathbf{v}$ in both, space and time [OF02]. Eulerian integration schemes compute s at a time step t_i from s and $\mathbf{v}$ at the time t_{i-1} regarding only specific locations, e.g., grid points. Lagrangian schemes follow the trajectories of particles forward over all time steps, while Semi-Lagrangian integration techniques evaluate s at time

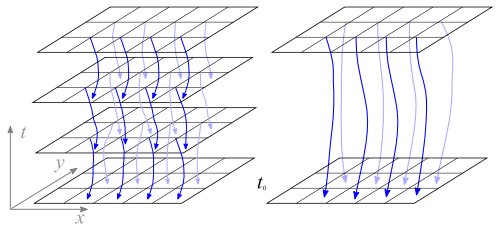


Figure 1: *Semi-Lagrangian (left) versus backward Lagrangian scheme (right). Both schemes update s at grid points by integrating backward in time. The Semi-Lagrangian scheme, however, does this for each step. In our setting we can apply a fully backward Lagrangian scheme, which requires only a single evaluation (interpolation) of s at t_0. The scheme is simpler, more efficient and more accurate.*

t_i by a single backward integration step of $\mathbf{v}$. Furthermore, there are hybrid schemes incorporating particle integrations to correct errors in the (Eulerian) integration of the PDE (1).

Eulerian schemes often suffer from stability problems as they are only conditionally stable. A common problem with fully Lagrangian schemes is a faithful reconstruction of s since the final particle distribution may be highly non-uniform. This is why Semi-Lagrangian schemes are often preferred (e.g., in fluid simulation [Sta99]). They ensure reconstruction by choosing grid points as evaluation points thus reverting to the spatial grid after each time step.

For our application we can rely on a much simpler integration scheme: a backward Lagrangian integration. Note that standard methods to solve (1) coming from level set theory and numerical flow simulation often assume that $\mathbf{v}$ and s are *not* independent. In fact, usually the definition of $\mathbf{v}$ incorporates local components of s such as its gradient, Hessian, or Gaussian and mean curvature of its isosurfaces, leading to the fact that $\mathbf{v}$ at a time t_i is not known until s has been computed in t_i. This is not the case for our approach: we define $\mathbf{v}$ *independently* of s. This allows for using a backward Lagrangian scheme to solve (1): the scalar value at a time t is obtained by a complete pathline integration back until t_0:

$$s(\mathbf{x},t) = s_0(\phi_t^{t_0}(\mathbf{x})). \tag{2}$$

This is illustrated in Figure 1. Note that for computing $s(\mathbf{x},t)$, it is not necessary to compute s at any intermediate time steps between t_0 and t. There are two main benefits of the backward Lagrangian scheme: firstly, improved accuracy as we do not suffer from interpolation artifacts that occur for a Semi-Lagrangian scheme – the scalar field s is evaluated only once at t_0. Secondly, integration involves fewer data and fewer operations and can be implemented more efficiently.

3.1. Properties of the deformation

Let $\mathbf{v}$ be a C^1 continuous vector field over D with the following properties:

- *local support*: $\mathbf{v}$ is non-zero only in some inner region of D (it is constantly zero at the boundary of D);
- *boundedness*: $\|\mathbf{v}\|$ and $\|\nabla\mathbf{v}\|$ do not diverge to infinity at any location in D;
- $\mathbf{v}$ is *divergence-free*.

Then the deformation s defined by (2) has the following properties:

(a) s is volume preserving: the volume inside *every* isosurface remains constant under the deformation;
(b) s is continuity preserving: if s_0 is C^1 continuous then s is C^1 as well.
(c) s is topology preserving: no isosurface changes its topology during the deformation.

Property (a) follows directly from the definition of divergence of vector fields [Dav67]. Property (b) has been proven in [vFTS06] for explicit surfaces, the same proof holds for implicit surfaces as well.

Regarding property (c), we realize that a topology change requires a critical point of s, i.e., we can rephrase this property as follows: no critical points can appear or disappear during the deformation. All critical points of s are obtained by integrating the critical points of s_0. We note that a point is a *critical point* iff ∇s is vanishing. In order to prove property (c) we observe how ∇s is changing under integration of $\mathbf{v}$ over time:

$$\frac{\partial(\nabla s)}{\partial\tau} = \lim_{\tau\to t}\frac{\nabla s(\phi_t^\tau(\mathbf{x}),\tau) - \nabla s(\mathbf{x},t)}{\tau - t} = \frac{\partial(\nabla s)}{\partial t} + H(s)\,\mathbf{v}\,, \tag{3}$$

where $H(s)$ denotes the Hessian of s. We rewrite the PDE (1) in matrix notation as

$$\begin{pmatrix} (\nabla s)^T & \frac{\partial s}{\partial t} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} = 0\,. \tag{4}$$

Then computing the gradient of (4) by applying the product rule gives

$$\begin{pmatrix} H(s) & \frac{\partial(\nabla s)}{\partial t} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} + \begin{pmatrix} \nabla(\mathbf{v}^T) & \mathbf{0} \end{pmatrix}\begin{pmatrix} \nabla s \\ \frac{\partial s}{\partial t} \end{pmatrix} = \mathbf{0}$$

where $\nabla(\mathbf{v}^T)$ is the transposed Jacobian matrix of $\mathbf{v}$. Evaluation of the terms and comparison with (3) yields

$$\frac{\partial(\nabla s)}{\partial\tau} = -\nabla(\mathbf{v}^T)\cdot\nabla s. \tag{5}$$

Equation (5) states that if we are at a critical point of s (i.e., $\nabla s = \mathbf{0}$), it remains a critical point under the integration of $\mathbf{v}$ (i.e., $\frac{\partial(\nabla s)}{\partial\tau} = \mathbf{0}$). Conversely, for a non-critical point, ∇s cannot vanish during the integration of $\mathbf{v}$ as otherwise a backward integration starting from the critical point would violate the previous statement.

3.2. Defining the vector field v

The divergence-free vector field $\mathbf{v}$ steers the deformation. Its definition is not a contribution of this paper, since we use

the method presented in [vFTS06] which defines deformations of explicit shapes (represented as triangular meshes) by vector field integration. For the sake of completeness we provide a brief review: essentially, the definition of $\mathbf{v}$ is an interactive process, where $\mathbf{v}$ is defined by three scalar fields $e(\mathbf{x},t)$, $f(\mathbf{x},t)$, $r(\mathbf{x},t)$ and two thresholds r_i, r_o. The region field r and thresholds r_i and r_o define an inner region of full deformation, a blended intermediate region, and a region of zero deformation. The full deformation is defined by scalar fields p and q as

$$\mathbf{v} = \nabla p \times \nabla q$$

with

$$p(\mathbf{x}) = \begin{cases} e(\mathbf{x}) & \text{if } r(\mathbf{x}) \leq r_i \\ (1-b)\,e(\mathbf{x}) & \text{if } r_i < r(\mathbf{x}) \leq r_o \\ 0 & \text{if } r_o < r(\mathbf{x}) \end{cases},$$

$$q(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } r(\mathbf{x}) \leq r_i \\ (1-b)\,f(\mathbf{x}) & \text{if } r_i < r(\mathbf{x}) \leq r_o \\ 0 & \text{if } r_o < r(\mathbf{x}) \end{cases},$$

and $b = b(r(\mathbf{x}))$ is a polynomial blending function with $b(r_i) = 0$, $b(r_o) = 1$ and $\frac{db}{dr}(r_i) = \frac{db}{dr}(r_o) = \frac{d^2b}{dr^2}(r_i) = 0$ represented in Bernstein-Bézier form. Vector fields $\mathbf{v}$ constructed this way are guaranteed to be divergence-free.

3.3. Deformation types

With the choice of the scalar fields e, f, and r we can define different types of deformations. Since our approach does not focus on a particular isosurface, e, f, r should be chosen to act on a family of isosurfaces simultaneously. For this purpose we adapt the approach presented in [vFTS07], which uses a spatial curve that guides the deformation, to our setting of deforming a family of implicit surfaces: the user specifies $r(\mathbf{x},t_0), r_i, r_o$ and a curve $\mathbf{p}(t)$ with $r(\mathbf{p}(t_0),t_0) \leq r_i$ interactively. The region field r and threshold r_i, r_o define the regions of full deformations, and $\mathbf{p}$ describes the path of the inner region over time. This can be imagined as sweeping a deformation tool with local support along $\mathbf{p}$. Defining $\mathbf{N}(t) = (\mathbf{t}(t), \mathbf{n}(t), \mathbf{b}(t))$ as the moving normalized Frenet frame of $\mathbf{p}$, we get

$$r(\mathbf{x},t) \;=\; r(\,\mathbf{p}(t_0) + \mathbf{N}(t_0)\,\mathbf{N}(t)^T\,(\mathbf{x}-\mathbf{p}(t))\,,\,t_0)\,,$$

and e, f describe a translation in the direction $\dot{\mathbf{p}}$:

$$e = \mathbf{n}^T(\mathbf{x}-\mathbf{p}) \quad \text{and} \quad f = ||\dot{\mathbf{p}}||\,\mathbf{b}^T(\mathbf{x}-\mathbf{p}).$$

Optionally, we allow a rotation around an axis given by a center $\mathbf{c}$ and a direction $\mathbf{d}$:

$$e = \mathbf{d}^T(\mathbf{x}-\mathbf{c}) \quad \text{and} \quad f = ||\mathbf{d}\times(\mathbf{x}-\mathbf{c})||^2\,.$$

Figure 2 illustrates two examples: a translation in the inner region following a curve $\mathbf{p}$ (left), and a subsequent rotation around an axis through the red center (right).

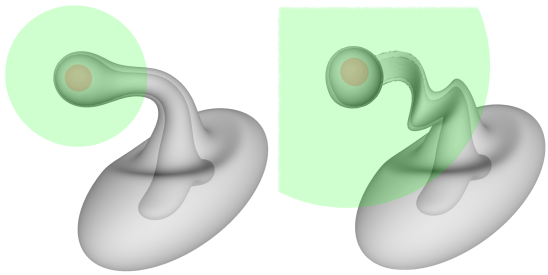


Figure 2: *Simple deformations. Example for translation along a curve (left) followed by multiple rotations (right). Inner region and intermediate region are colored red and green, respectively. For each case two deformed isosurfaces are shown.*

4. Implementation

In this Section we describe the implementation of our approach. Generally, we discretize scalar fields on uniform grids, and we use a tricubic C^1 interpolation for evaluation of function values and gradients. For pathline integration we apply a fourth order Runge-Kutta scheme with adaptive step size control (see, e.g., [PTVF07]).

4.1. User interface and real-time visualization

The user first loads an initial scalar field which is uploaded to the GPU. Then she can set parameters of the tool interactively, e.g., type of deformation, region fields, etc., and perform sweeps in real-time. A particular isosurface is displayed for interactive modeling, the isovalue can be varied freely. In our experiments this choice has shown to be more intuitive than a more general real-time volume rendering approach (see, e.g., [HKRW06] for an overview). The visualization uses a real-time GPU version of the marching cubes algorithm based on histogram pyramids [DZTS08], which does not show any significant impact on run time of the main algorithm in all our experiments. Marching cubes is performed only if either the scalar field or the isovalue changes, and the result is saved into a vertex buffer.

4.2. Interactive scalar field manipulation

In all our examples we use a grid resolution of 256^3. The maximum resolution is mainly limited by available memory as the whole field is required to persist in GPU memory for visualization. We remark that an out-of-core implementation of our general approach is straightforward, currently it is the marching cubes visualization which requires that all data persists on the GPU. For the deformation approach itself we use s and a temporary buffer with size of the region of interest on the GPU. Both, pathline integration and scalar field interpolation are performed by the GPU.

Deformations are specified as control points of the sweep curve $\mathbf{p}$ together with additional properties such as region

fields. Deformations can be extended by editing the current curves and adding new ones interactively, restriction provides an undo functionality. The associated parameter sets are sent to the GPU, their memory footprint is not significant. When evaluating $\mathbf{v}$, we exploit the fact that the individual operations have local support, i.e., $\mathbf{v}$ is non-zero only within a region of interest and can therefore be evaluated on a smaller grid.

For practical real-time editing we relax the integration scheme slightly using intermediate results: we partition a sweep along the guiding curve $\mathbf{p}$ (see Section 3.3) into time intervals, each of which describes a partial modification. From the arising series of modified scalar fields only the most recent scalar field is stored in a GPU buffer of constant size. This means that we do not integrate all way back to t_0 but only to the last interval bound. However, each edit still comprises a significant time interval and potentially many integration steps – we are not switching to a Semi-Lagrangian scheme. We use this compromise of buffering intermediate edits to balance accuracy and efficiency. This way we save on the integration process with shorter time intervals that only reach back to the previous partial edit, and we can guarantee real-time response. The price is a slight loss of accuracy due to interpolation of intermediate data. Of course we can control granularity of time intervals, and we can even invest some extra time and do a full pathline back integration to t_0 at any time to visualize a better approximation.

4.3. Offline high-quality isosurface extraction

The quality of the rendered isosurfaces depends on the grid resolution for the marching cubes algorithm. We use the same resolution as for computation of $s(t)$ and are generally limited by GPU memory. While this is acceptable for real-time visualization, it is evident that artifacts show up whenever the sampling rate becomes too low depending on the particular isovalue (see Figure 5 (a)). This is not a failure of our method but a failure of reconstruction. To show this we extract high-quality isosurfaces using the surface meshing routines provided by the CGAL library [CGA09]. We chose this library because the underlying adaptive meshing algorithm [BO05] is extremely robust and produces highly accurate triangulations of the isosurface. Even extreme examples with near overlapping surface parts are reconstructed faithfully, see Figure 5 (b-d). The high-quality isosurfaces are extracted at the cost of using a sequential CPU implementation that takes the recorded parameter sets as input and integrates the pathlines. We note that this is inherently a sequential process: the meshing algorithm does not have enough information of the surface and treats evaluation as a black-box component, the so called oracle. We do not have any influence on location and order of evaluation points, which seem highly non-uniform. As a consequence this is clearly an off-line process. We observed that sampling $s(t)$ on a uniform grid and using the CGAL surface mesher on this as input provides a significantly more efficient and fairly accurate reconstruction. However, this compromise is less exact and depends on the sampling rate.

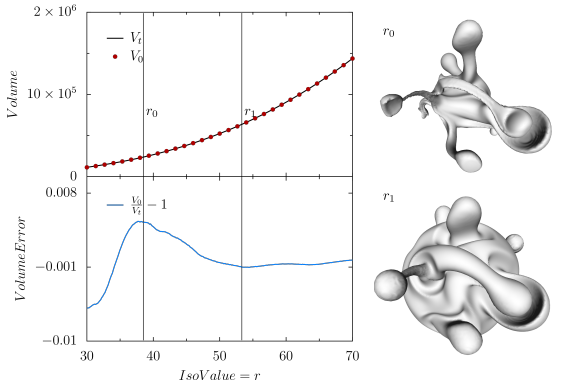


Figure 3: *Volume preservation for a randomly deformed sphere. The top figure shows the evaluation of the volume V_t of the deformed sphere and the original sphere ($V_0 = 4\pi k^3$) plotted over varying isovalues r (radii). The curves are nearly identical. The bottom figure shows the relative difference. Two particular isosurfaces of the deformed sphere are shown on the right for isovalues r_0 and r_1.*

5. Analysis

In this Section we analyze properties of our approach and its implementation in practice.

Volume Preservation. We compare the volume of deformed isosurfaces to ground truth. Even though the volumes are computed from meshes output by marching cubes our experiments confirm volume preservation. Figure 3 plots volume over isovalues for an initial sphere and a randomly deformed version (see isosurfaces corresponding to r_0, r_1). The relative error is low even though reconstruction artifacts come into play for smaller isovalues. Figure 4 shows a similar experiment for few isovalues and again surface triangulations from marching cubes for t_0 and two particular time steps. Again errors are not significant.

Accuracy of reconstruction. The static grid of the GPU implementation may lead to undersampling artifacts. For all our experiments, we could extract high-quality surfaces with correct topology and without any self-intersections. Figure 5 compares marching cubes to reconstruction with [BO05] and shows extreme configurations with regions of the isosurfaces getting very close to each other. The volume variation of all shown high-quality reconstructions was always below 0.01%.

Topology preservation. Figure 6 shows two isosurfaces for an extreme deformation of a double torus. The isosurfaces

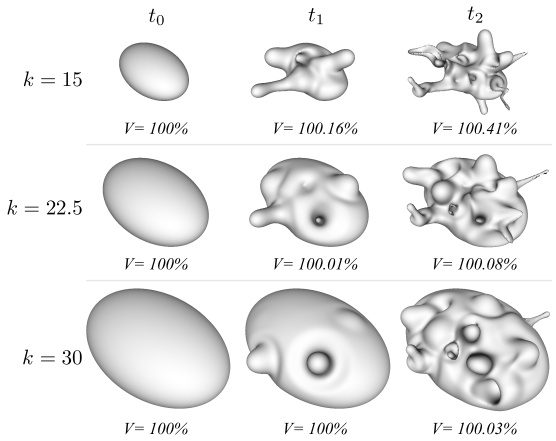


Figure 4: *Volume preservation. Three isosurfaces for isovalues k are shown at time t_0,t_1,t_2 of an interactive modeling session and marching cubes reconstruction. The variation of their volume V is low even for extreme deformations.*

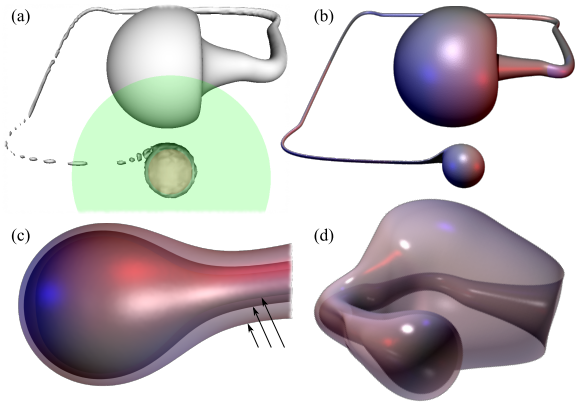
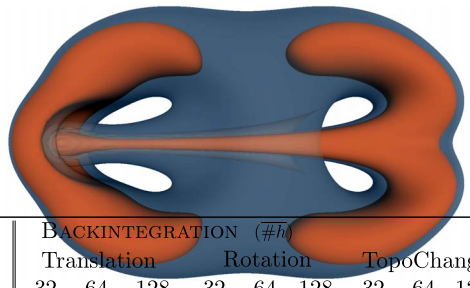


Figure 5: *Comparison of different isosurface reconstructions. (a) An extreme deformation leads to artifacts for marching cubes reconstructions during interactive modeling. (b) The same surface reconstructed exactly. Two (d) and three (c) interfaces getting close to each other are reconstructed correctly.*

have different topology – a single genus-2 surface and two disconnected genus-0 surfaces – at t_0, and their topology is preserved over the whole deformation. In fact, topology preservation proved in Section 3.1 was confirmed by all our experiments (disregarding artifacts from marching cubes).

Performance. We implemented our approach using NVIDIA's CUDA interface. All timings are measured for a NVIDIA GTX280 GPU with 1GB memory and an AMD Opteron processor at 2.6GHz. Figure 7 summarizes integration and interpolation timings. The processing time of cubic interpolation is negligible. Numerical pathline integration is the most time consuming part of our algorithm. The average number of adaptive integration steps is 64 in all our examples. Typically we observe numbers of seven (Figure 5 (a)) up to 250 (Figure 8 (right)) time steps when using intermediate scalar fields. This confirms that even numbers of 10^6 pathlines can be handled efficiently. The main reason is that no time consuming vector field lookup into texture space is required as our vector fields are parameterized and evaluated in closed form. Finally, we remark that a full backward integration is more expensive and corresponds to a sum over all time steps back to t_0, for most of our examples this is in the order of 1000 steps in total. A single marching cubes surface reconstruction took 14*ms* on average depending on the number of occupied voxels.



	BACKINTEGRATION ($\#h$)								
	Translation			Rotation			TopoChange		
$\lvert X\rvert$	32	64	128	32	64	128	32	64	128
10^3									
10^4									
10^5									
10^6									

Figure 6: *Topology preservation. A double torus (transparent blue) was deformed from its right to its left handle. The deformation preserves its genus-2 topology. At the same time the topology of an inscribed unconnected isosurfaces with lower isovalues (solid red) remains unchanged. (Figure 5 (c) shows a comparable closeup of the blue isosurface using exact reconstruction.)*

$\bar{h}$ / N	Translation			Rotation			INTERPOLATION	
	32	64	128	32	64	128	Linear	Cubic
10^3	0.96	1.36	2.16	0.87	1.21	1.81	0.01	0.01
10^4	1.55	2.41	4.62	1.87	2.15	3.65	0.01	0.02
10^5	7.42	13.74	26.41	6.12	11.16	21.42	0.03	0.04
10^6	64.65	125.28	247.71	48.55	92.63	180.97	0.04	0.05

Figure 7: *Timings (in ms) for N GPU pathline integrations with $\bar{h}$ average integration steps each (left) and N scalar field interpolations (right).*

$\bar{h}$ / N	Translation			Rotation			TopoChange		
	32	64	128	32	64	128	32	64	128
10^3									
10^4									
10^5									
10^6									

6. Applications

Designing smooth surfaces from scratch. Our approach can be used for general modeling of smooth isosurfaces starting from a smooth field, e.g., representing a family of spheres. The volume preservation property mimics Plasticine-like materials plausibly and hence leads to a natural object behavior and intuitive editing. In addition, constant topology is guaranteed for correct results. As a remarkable feature these two properties are satisfied not only for a single

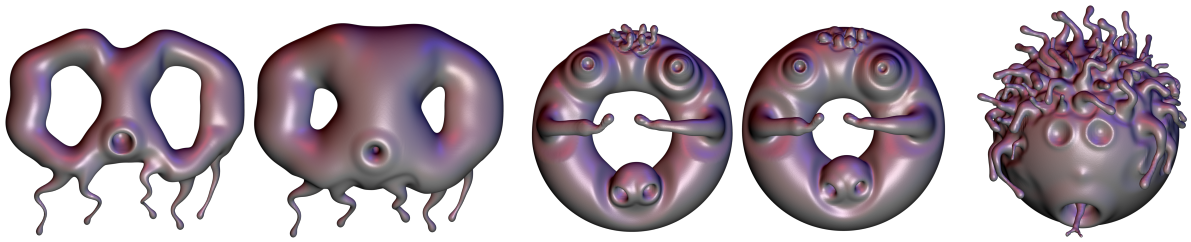


Figure 8: *Faces of different topology. Isosurfaces of three modified scalar fields. The left and center pairs of images show two isosurfaces of the same scalar field:* all *isosurfaces are deformed simultaneously while their volume and topology are preserved.*

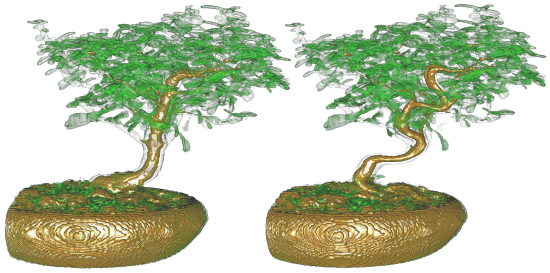


Figure 9: *Volume rendering of original (left) and deformed (right) bonsai data set.*

isosurface of interest but for all isosurfaces, i.e., the deformation acts on a family of surfaces. Figure 8 shows examples that were modeled in $3-20$ minutes by an inexperienced user.

Interactive deformation of volume data. Volume preserving deformations of a single isosurface result in modifications of all isosurfaces within the inner and blended region fields. Figure 9 shows a deformation of the bonsai data set, where the tool was swept near the trunk.

7. Discussion

In this Section we give a comparison of our approach to existing ones.

There are a number of approaches for deforming implicit surfaces, among them some aiming at the preservation of the volume inside a particular isosurface [CA06]. However, to the best of our knowledge, our approach is the first one which preserves the volume of all of isosurfaces. Also, topology preservation of all isosurfaces is not addressed so far.

In comparison to other real-time volume deformation techniques [RSSG01, WR01, SBH07] our approach allows for stronger and more localized deformations, also volume and topology preservation is not addressed there.

The approaches closest to us are [vFTS06, vFTS07] which define deformations by vector field integration as well. Contrary to us, they work only on one explicit surface, and they apply a forward integration of mesh vertices.

In volume modeling, a popular classification of approaches distinguishes physically based and non-physically based. Our approach lies between these classes: although it is not explicitly physically based, volume preservation is a physically justified property for many materials.

In comparison to well-established integration techniques for level sets, our approach applies a backward Lagrangian integration scheme which cannot be applied to general level sets due to the dependencies of the level sets and the steering field.

Our approach has the following limitations:

Although the volume preservation appears to be natural condition for plausible deformations, there are applications where volume preservation does not hold. For example, the growing of a tumor in a medical volume data set cannot be addressed by our approach.

Since we consider the whole volume, well-established deformation tools which focus on a particular surface do not fit into our approach. This includes sculpting techniques like cutting or carving [PF01] as well as volumetric copy and paste techniques [MBWB02].

Since our technique only incorporates the volume of the isosurfaces, metric distortion of an isosurface is not addressed. Moreover, analytically represented implicit surfaces can not be deformed exactly by the integration scheme without discretizing the defining function.

The limitations mentioned above are inherent to our approach. In addition, there are algorithmic limitations which can be addressed in future research: the fixed grid resolution of the volume data is a compromise between interactive performance and accuracy and is limited by GPU memory. This also naturally limits the spacial extents of the domain, and artifacts may appear for isosurfaces near grid boundaries unless special care is taken. Additionally due to reconstruction artifacts topology preservation may appear corrupted for real-time visualization. Hardware accelerated adaptive grids are a candidate for improvement here. Finally, more advanced deformation tools can be developed. In principle, different choices of the fields e, f, r which stay independent of the underlying scalar field allow for customized tools, but this depends on and has to be set for different application scenarios.

Acknowledgements

This work was supported by the German National Academic Foundation.

References

[ACWK04] ANGELIDIS A., CANI M.-P., WYVILL G., KING S.: Swirling-sweepers: Constant volume modeling. In *Pacific Graphics* (2004). 2

[AS05] ALEXANDROV O., SANTOSA F.: A topology-preserving level set method for shape optimization. *Journal of Computational Physics 204* (2005), 121–130. 2

[AWC04] ANGELIDIS A., WYVILL G., CANI M.-P.: Sweepers: Swept user-defined tools for modeling by deformation. In *Shape Modeling International* (2004), pp. 63–73. 2

[BO05] BOISSONNAT J.-D., OUDOT S.: Provably good sampling and meshing of surfaces. *Graph. Models 67*, 5 (2005), 405–451. 5

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 213–230. 2

[CA06] CANI M.-P., ANGELIDIS A.: Towards virtual clay. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), pp. 67–83. 2, 7

[CCI*07] CHEN M., CORREA C. D., ISLAM S., JONES M. W., SHEN P.-Y., SILVER D., WALTON S. J., WILLIS P. J.: Manipulating, deforming and animating sampled object representations. *omputer Graphics Forum 26*, 4 (2007), 824–852. 2

[CGA09] CGAL: Computational Geometry Algorithms Library, 2009. http://www.cgal.org. 5

[Coh09] COHEN-OR D.: Space deformations, surface deformations and the opportunities in-between. *Journal of Computer Science and Technology 24*, 1 (2009), 2–5. 2

[Dav67] DAVIS H.: *Introduction to vector analysis*. Allyn and Bacon, Inc., Boston, 1967. 3

[DCG98] DESBRUN M., CANI-GASCUEL M.-P.: Active implicit surface for animation. In *Graphics Interface* (1998), pp. 143–150. 2

[DG95] DESBRUN M., GASCUEL M.: Animating soft substances with implicit surfaces. In *Proc. SIGGRAPH* (1995), pp. 287–290. 2

[DZTS08] DYKEN C., ZIEGLER G., THEOBALT C., SEIDEL H.-P.: High-speed marching cubes using histopyramids. *Computer Graphics Forum 27*, 8 (2008), 2028–2039. 4

[FPRJ00] FRISKEN S. F., PERRY R. N., ROCKWOOD A. P., JONES T. R.: Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proc. SIGGRAPH 2000* (2000), pp. 249–254. 2

[GB08] GAIN J., BECHMANN D.: A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics 27* (2008), 1–21. 2

[Gib97] GIBSON S. F. F.: 3D chainmail: A fast algorithm for deforming volumetric objects. In *SI3D* (1997), pp. 149–154, 195. 2

[GW06] GEORGII J., WESTERMANN R.: A multigrid framework for real-time simulation of deformable bodies. *Computers & Graphics 30*, 3 (2006), 408–415. 2

[HKRW06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D.: *Real-time Volume Graphics*, 1 ed. A K Peters, July 2006. 2, 4

[MBWB02] MUSETH K., BREEN D. E., WHITAKER R. T., BARR A. H.: Level set surface editing operators. In *Proc. SIGGRAPH* (2002), pp. 330–338. 2, 7

[MJBF02] MILLIRON T., JENSEN R. J., BARZEL R., FINKELSTEIN A.: A framework for geometric warps and deformations. *ACM Transactions on Graphics 21*, 1 (2002), 20–51. 2

[MQW01] MCDONNELL K. T., QIN H., WLODARCZYK R. A.: Virtual clay: a real-time sculpting system with haptic toolkits. In *Proc. I3D* (2001), pp. 179–190. 2

[OF01] OSHER S., FEDKIW R. P.: Level set methods: An overview and some recent results. *Journal of Computational Physics 169* (2001), 463–502. 2

[OF02] OSHER S. J., FEDKIW R. P.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002. 2

[PF01] PERRY R. N., FRISKEN S. F.: Kizamu: A system for sculpting digital characters. In *Proc. SIGGRAPH* (2001), pp. 47–56. 2, 7

[PTVF07] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007. 4

[RSSG01] REZK-SALAMA C., SCHEUERING M., SOZA G., GREINER G.: Fast volumetric deformation on general purpose hardware. In *Graphics Hardware* (2001). 2, 7

[RWE08] RÖSSLER F., WOLFF T., ERTL T.: Direct GPU-based volume deformation. In *Proc. CURAC* (2008), pp. 65–68. 2

[SBH07] SCHULZE F., BÜHLER K., HADWIGER M.: Interactive deformation and visualization of large volume datasets. In *Proc. GRAPP* (2007), Braz J., Vázquez P.-P., Pereira J. M., (Eds.), pp. 39–46. 2, 7

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *Proc. SIGGRAPH* (1986), vol. 20, pp. 151–160. 2

[Sta99] STAM J.: Stable fluids. In *Proc. SIGGRAPH* (1999), pp. 121–128. 3

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. In *Proc. SIGGRAPH* (2006), pp. 1118–1125. 2, 3, 4, 7

[vFTS07] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Explicit control of vector field based shape deformations. In *Proc. Pacific Graphics* (2007), pp. 291–300. 2, 4, 7

[WR01] WESTERMANN R., REZK-SALAMA C.: Real-Time Volume Deformation. *Computer Graphics Forum (Proc. Eurographics) 20*, 3 (2001). 2, 7