Spring Level Sets: A Deformable Model Representation to Provide Interoperability between Meshes and Level Sets

Blake C. Lucas, Michael Kazhdan, and Russell H. Taylor, *Fellow*, *IEEE*

Abstract—A new type of deformable model is presented that merges meshes and level sets into one representation to provide interoperability between methods designed for either. This includes the ability to circumvent the CFL time step restriction for methods that require large step sizes. The key idea is to couple a constellation of disconnected triangular surface elements (springls) with a level set that tracks the moving constellation. The target application for Spring Level Sets (SpringLS) is to implement comprehensive imaging pipelines that require a mixture of deformable model representations to achieve the best performance. We demonstrate how to implement key components of a comprehensive imaging pipeline with SpringLS, including image segmentation, registration, tracking, and atlasing.

Index Terms—Segmentation, registration, tracking, atlas, shape model

1 Introduction

Deformable models are geometric representations of objects that deform (change shape) due to forces applied at their boundary. Deformable models are applicable to a broad range of problems in image analysis, computer vision, and computer graphics including: reconstruction, nonrigid registration, image segmentation, atlasing, animation, constructive solid geometry, fluid simulation, and motion tracking. There are two major model representations: meshes [1] and level sets [2], [3]. Deformable model methods usually favor a particular representation (i.e., meshes for nonrigid registration and level sets for image segmentation). However, large systems that use a mixture of methods are forced to transform one representation into another in order to use the preferred representation for each method. This strategy leads to loss of information and less flexibility in design of the system.

One incarnation of a fully automated imaging pipeline for localization, registration, segmentation, tracking, and atlas construction is depicted in Fig. 1. Systems that implement large sections of a Comprehensive Imaging Pipeline (CIP) are emerging in the literature. In the work by Kohlberger et al. [4], they begin with machine learning followed by mesh-based atlas registration and conclude with level set active contour segmentation. In the transition from mesh to level set representation, they lose point correspondences between the deformable model and atlas. This impedes their ability to progress further through the CIP and refine their atlas. By comparison, Tosun et al. [5] implement the second half of the CIP. They begin with an active contour segmentation of the human brain cortex and then parametrically map the isosurface to an atlas. The drawback of their system is that the parametric atlas they construct cannot be fed back into their active contour segmentation, which uses an implicit level set representation.

This work presents a new type of deformable model that can be used to implement all stages of the CIP. The key idea is to couple a constellation of disconnected triangular surface elements (springls) with a level set that tracks the moving constellation. Attractive forces between springls hold the constellation together, and methods are provided for resampling the constellation as the model deforms. Spring Level Sets (SpringLS) can be interpreted as a mesh or level set. The representation is intended to be used for methods that employ a mixture of mesh and level set techniques, but it is applicable to almost all deformable model methods. SpringLS was first introduced in conference form [7] and is extended in this work. Source code is available at http://code.google.com/p/imagesci/.

2 Related Work

2.1 Meshes

Meshes were the earliest representation for deformable models [1]. In this framework, the model is deformed by perturbing mesh vertices. The model's boundary is explicitly tracked by remembering the trajectory of each vertex. As a section of the mesh expands or contracts, sharp creases, edges, self-intersections, or triangle flips can develop. Sharp edges and other mesh artifacts violate a common material property that objects represented by deformable models are between elastic and plastic on the

- *B.C. Lucas is with the JHU Applied Physics Laboratory and the Department of Computer Science, Johns Hopkins University, 112 Hackerman Hall, 3400 N. Charles Street, Baltimore, MD 21218. E-mail: blake@cs.jhu.edu.*
- *M. Kazhdan and R.H. Taylor are with the Department of Computer Science, Johns Hopkins University, 112 Hackerman Hall, 3400 N. Charles Street, Baltimore, MD 21218. E-mail: misha@cs.jhu.edu, rht@jhu.edu.*

Manuscript received 29 Feb. 2012; revised 16 July 2012; accepted 20 July 2012; published online 25 July 2012.
Recommended for acceptance by H. Pottmann.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-02-0044.
Digital Object Identifier no. 10.1109/TVCG.2012.162.

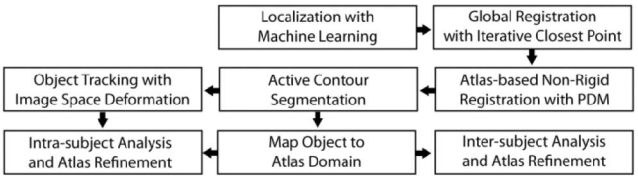


Fig. 1. Concept of a comprehensive imaging pipeline that localizes, registers, segments, and tracks objects. It then uses the segmented objects to iteratively refine the atlas and analyze intra/intersubject variability of geometric structures. PDM refers to point distribution model [6], although any parametric atlas representation could be used in its place.

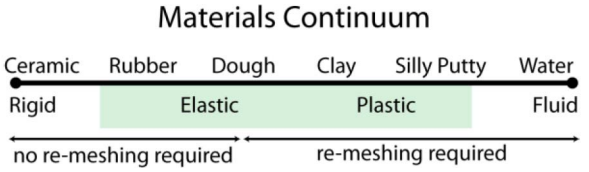


Fig. 2. The materials continuum [14]. In imaging applications, deformable models imitate the behavior of materials between elastic and plastic (shown in green).

materials continuum (Fig. 2). To reduce artifacts, the mesh must be regularized and resampled (remeshed) periodically. Remeshing is challenging (see Wojtan et al. [8] for an in-depth discussion) because triangles must be connected to form a watertight model [9], [10], [11], [12], [13], and remeshing interferes with vertex tracking in nonobvious ways. These challenges have motivated the creation of new deformable model representations (e.g., marker level sets (MLS) and surfels) that do not require remeshing, handle topology changes easily, cannot self-intersect, and track moving points instead of mesh vertices.

2.2 Level Sets

The level set method (LSM) [2], [3] represents a deformable model as a 3D image where the image intensity at each voxel is the distance to the surface of the object. Distance measurements are signed: negative values are inside and positive values are outside the object. A triangle mesh can be extracted by computing the iso-surface corresponding to the zero level set of the image. The level set representation has several advantages over deformable meshes: 1) no need for self-intersection removal; 2) topology change is easy; 3) no need to remesh. These properties have made level sets the popular choice for image segmentation and fluid-like nonrigid deformation.

Some recent remeshing schemes [8], [10], [15] construct a level set to aid in remeshing. Note that since the watertight property is provided by the level set in these methods, it may not be necessary to have a watertight mesh. This observation is one motivation for Spring Level Sets.

Level sets are difficult to use for registration and tracking tasks because there is no innate ability to track vertices as in the mesh deformation framework. The surface only exists when an iso-surface is extracted from the level set. Furthermore, the level set is stored as an image that is resampled at each time step. Resampling an image acts as a low-pass filter that results in feature loss as a function of the number of time steps, even if the motion is rigid (i.e., global registration) or divergence free.

2.3 Other Representations

There are other deformable model paradigms (see Nealen et al. [16] for a survey of physically-based deformable models). Model selection is motivated by properties of the material being modeled, and there is no representation that is superior for modeling all materials. Different phases of imaging pipelines require the model to behave as different types of materials, such as: a rigid solid during registration, elastic solid during nonrigid registration, and plastic during segmentation. This leads to imaging pipelines with transformations between model representations. Attempts have been made to unify deformable model representations with varying success through either volumetric [17], [18], [19], [20], [21] or surface [11], [22], [23], [24], [25] approaches. One of the main challenges these works address is how to resample or remesh the model during deformation, given their representation. Surface elements are more appropriate for imaging problems because objects are, by definition, "homogeneous" in some sense and differentiated by variations in image intensities that occur near their boundaries. Of these surface-based representations, the Marker Level Set [24] and surfels [25] are closest to this work.

2.3.1 Marker Level Set (MarkerLS)

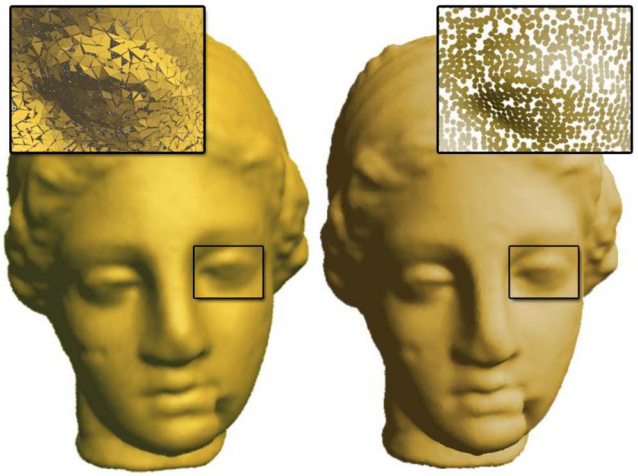
The Marker Level Set method [24] maintains a set of particles located on the level set's zero iso-level. Since particles lie exactly on the zero iso-level, they can be used to track the model's boundary. After each level set and particle advection step, the level set is corrected so that particles continue to lie on the level set's zero iso-level. Particles are added to cover the zero iso-level and deleted to prevent oversampling. The MarkerLS method associates a label (e.g., RGB color) with each particle to encode tracking information. When a new particle is added, it is assigned a label by interpolating between labels of its neighbors.

The main difference between SpringLS and MarkerLS is that springls define the model's level set, whereas MarkerLS uses particles to correct errors in the level set. MarkerLS requires the deformation method to have an equivalent level set and parametric interpretation in order to deform both representations. Also, it is not appropriate for deformations that require large time steps (e.g., elastic deformations) because the level set method's time step size is limited by the CFL condition.

Movement of the auxiliary level set with SpringLS is passive and independent of the deformation method. SpringLS can be applied more broadly to deformation methods that require large step sizes and methods that only have a parametric interpretation (e.g., Point Distribution Models [6]).

2.3.2 Surfels

Systems for point-sampled shape modeling [25], [26] also relate to this work. In these systems, a manifold surface is approximated by a set of points or disk-shaped surface elements (surfels); from which, a watertight surface can be inferred using Moving Least Squares [27]. The model is deformed by perturbing or stretching surfels. The deformation process creates holes and irregular sampling patterns in the model's representation. To mitigate these problems, several filtering operations are performed. To prevent



(a) (b)

Fig. 3. Igea model represented by a constellation of (a) springls and (b) surfels. Springls were rendered with volumetric raycasting and surfels were rendered with Pointshop3D.[1] Insets show the constellation of surface elements. The constellation in (a) was reexpressed as surfels to use in the rendering for (b). Consequently, the sampling pattern and number of surface elements is the same.

overstretching, elongated surfels are split in two; to regularize the sampling distribution, neighboring surfels repel each other; and to fill holes, surfels are inserted. Filtering assumes that the object is smooth, and therefore neighboring surfels can be projected onto a tangent plane with minimal geometric distortion.

Both point-sampled modeling and SpringLS have an auxiliary implicit representation; however, a level set has more utility in imaging applications than an MLS surface because a level set maintains an inside/outside assignment for each voxel in an image at all times during the deformation, which is important for determining and enforcing a model's topology. This is also true of level set variants that maintain particles [23], [24], [28].

Point-sampled modeling and SpringLS differ in their choice of surface element (Fig. 3). SpringLS use triangular surface elements (Fig. 3a inset) instead of disk surface elements (Fig. 3b inset) to prevent the loss of shape information in the transformation from triangle mesh into springls. Methods to approximate triangle meshes or level sets with surfels are challenging [29], [30], [31] because they require the introduction of more elements or a new type of element (i.e., clipped surfels as described by Pauly et al. [25]) to express regions with high curvature. SpringLS can express high curvature regions without approximation.

2.3.3 Complete Distance Field Representation (CDFR)

The Complete Distance Field Representation [32] is related to Spring Level Sets because CDFR maintains a dual level set and triangle mesh representation of an object. Instead of just interpolating the level set image, CDFR computes the exact distance to the triangle mesh when measuring distances close to the level set's zero iso-surface. The benefit of CDFR is that the model's accuracy is not limited by the level set's grid resolution, which is important when rendering objects with sharp edges, corners, or fine detail.

1. http://graphics.ethz.ch/pointshop3d/.

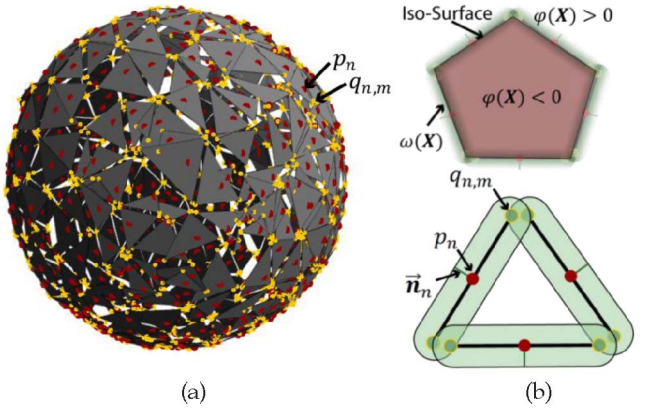


Fig. 4. (a) Springl constellation in 3D with particles (red), vertices (yellow), and surface elements (gray). (b) Signed level set and springl constellation in 2D.

3 Overview

In this work, we apply Spring Level Sets to tasks that require the deformable model to behave as three different types of materials: rigid, elastic, and plastic. Our focus will be on the deformable model's performance on imaging tasks. We will describe three comprehensive applications that require a mixture of deformable model methods as in the CIP (Fig. 1). These include atlas-based pelvis segmentation, motion tracking in 4D CT, and morphing of articulated characters. We begin by describing the SpringLS representation.

4 Method

4.1 Representation

A springl is a triangular surface element consisting of a particle and three springs connecting the particle to each of the triangle's vertices (see Fig. 4a).

A springl (S_n) is represented as five points describing the particle location p_n, correspondence point location a_n, and triangle vertices $q_{n,m}$. A correspondence point is a mapping between p_n on the current surface and a point a_n on the original surface. The clamped unsigned distance function $d_n(\boldsymbol{X})$ is computed and also associated with each springl. These distance functions are used to form the unsigned level set $\omega(X) = \min_n d_n(\boldsymbol{X})$. The support of a springl is represented by a capsule (green region in Fig. 4b). Springs, vertices, and particles are coplanar, and the angles between springs are fixed. An auxiliary signed level set $\varphi(\boldsymbol{X})$ is maintained and evolved with the particles (Fig. 4b). The level set augments the particle representation in several ways: 1) the signed distance function indicates regions that are inside or outside the model; 2) the signed level set indicates when new springls need to be added or removed; 3) the iso-surface extracted from the level set is a watertight triangle mesh representation of the model. The deformable model is stored as three data structures: a triangle soup representing surface elements ($q_{n,m}$), a point cloud of particles (p_n), and a 3D image representing the signed level set ($\varphi(\boldsymbol{X})$). Table 1 defines terms and expressions used for Spring Level Sets, and Table 2 summarizes the core algorithms that will be described in subsequent sections.

TABLE 1
Definition of Terms and Expressions

Symbol	Definition
p_n	Particle location.
a_n	Corresponding particle location on initial surface.
$q_{n,m}$	Triangle vertex $m = \{0,1,2\}$.
$\vec{n}_n$	Triangle normal.
S_n	4x4 matrix describing a springl.
$q_{n,m,k}$	k^{th} closest point on nearby triangle edge from $q_{n,m}$
d_{max}=0.5	Springl capsule radius.
$r = 0.05$	Particle and triangle vertex radius.
$R = d_{max} + 2r$	Nearest-neighbor range.
$\omega(X)$	Unsigned level set.
$\varphi(X)$	Signed level set.
$\varphi_{ref}(X)$	Initial signed level set.
$\alpha(x)$	Weight function.
$\beta(x)$	Threshold function.
λ_c	Threshold function smoothness.
λ_ρ	Pressure force weight.
λ_σ	Advection force weight.
λ_A	Affine transformation weight.
λ	Level set regularization weight.
κ_{int} and κ_{ext}	Spring constants for internal and external forces.
$\xi = \varrho = 0.1$	Step sizes for relaxation.
$M(\vec{b}_n)$	Rotation matrix representing the moment $\vec{b}_n$.
$VE(S_n)$	Local potential energy of springl n.
$\mathbb{I}_{\{x \le y\}}$	Function that is 1 for $x \le y$ and 0 otherwise.

TABLE 2
Description of Algorithms

Algorithm	Function
Advect	Apply forces to move springls.
Relax	Regularize springls constellation.
Contract	Remove poor quality springls and those far from the signed level set's iso-surface.
Resample	Split springls whose area exceeds a threshold.
FillGaps	Add more springls to fill holes in the constellation.
Evolve	Evolve signed level set to track springls.
Rebuild	Rebuild the signed distance field.

4.2 Advection

The model is deformed by incrementally advecting springls with Lagrangian methods (**Advect**). The deformation may be driven by pressure $\rho(\cdot)$ in the normal direction $\vec{n}_n$ of a springl, an external velocity field $\vec{\sigma}(\cdot)$, and/or a local affine transformation $\mathbf{A}(\cdot)$ that is a function of the particle position and its corresponding point a_n on the initial object:

$$\frac{\partial p_n}{\partial t}(t) = \lambda_\rho \vec{n}_n \rho(p_n(t)) + \lambda_\sigma \vec{\sigma}(p_n(t)) + \lambda_A \mathbf{A}(p_n(t), a_n) p_n(t), \quad (1)$$

and

$$\frac{\partial q_{n,m}}{\partial t}(t) = \lambda_\rho \vec{n}_n \rho(p_n(t)) + \lambda_\sigma \vec{\sigma}(p_n(t)) + \lambda_A \mathbf{A}(p_n(t), a_n) q_{n,m}(t). \quad (2)$$

See Table 1 for definitions of terms. The advection equations ((1) and (2)) are sufficient for modeling a broad spectrum of materials, and note that vertices are advected based on velocities evaluated at the particle location. Fluid deformation methods use pressure (ρ) and external velocity field ($\vec{\sigma}$), whereas rigid and elastic deformation methods use local affine transformations ($\mathbf{A}$).

4.3 Relaxation

4.3.1 Spring-Mass System

After each advection step, particles are fixed and the shape/orientation of springls are adjusted. To understand the relaxation process, we first introduce the system of forces as a spring-mass system.

Springls are only permitted to rotate around their particle and adjust their spring lengths, so we reexpress vertices in a local coordinate system as $\tilde{q}_{n,m} = q_{n,m} - p_n$, spring lengths $l_{n,m} = \|\tilde{q}_{n,m}\|$, and tangent vectors $\vec{s}_{n,m} = \frac{1}{l_{n,m}} M(-\vec{b}_n)\tilde{q}_{n,m}$. A springl configuration can be transformed back into world coordinates through

$$q_{n,m} = l_{n,m} M(\vec{b}_n)\vec{s}_{n,m} + p_n. \quad (3)$$

$M(\vec{b}_n)$ is a rotation matrix corresponding to the springl's pose, represented by an axis-angle vector $\vec{b}_n \in \mathbb{R}^3$. $l_{n,m}$ and $\vec{b}_n$ express the six degrees of freedom each springl has during relaxation. Springls have an internal spring force $\vec{f}^{\text{int}}_{n,m}$ that attracts vertices along their tangent vectors $\vec{s}_{n,m}$ toward the particle p_n

$$\vec{f}^{\text{int}}_{n,m} = \kappa_{\text{int}}(l_{n,m} - 2r) M(\vec{b}_n)\vec{s}_{n,m}, \quad (4)$$

where κ_{int} is the internal springl constant. The resting spring length is chosen to be $2r$ to discourage triangle flips. Although not implemented in this work, a rigorous approach to correct flips would be to test if $\vec{n}_n \cdot \nabla\varphi(X) < 0$ and reorient springls accordingly. An external spring force $\vec{f}^{\text{ext}}_{n,m}$ attracts vertices to the edges of nearby triangles

$$\vec{f}^{\text{ext}}_{n,m} = \kappa_{\text{ext}} \sum_k (q_{n,m} - q_{n,m,k}), \quad (5)$$

where κ_{ext} is the external spring constant and $q_{n,m,k}$ is the closest point to $q_{n,m}$ on the perimeter of the kth neighboring springl within a radius R from particle p_n (Fig. 5). To detect whether a springl is within a radius of R from a vertex, only the particle location for the springl is checked. This was done for computational efficiency.

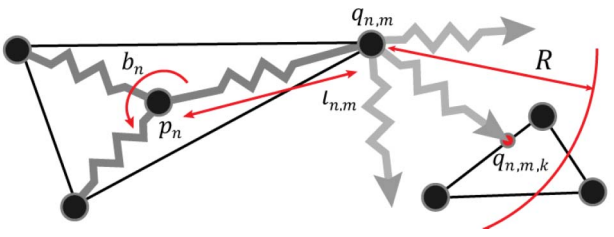


Fig. 5. Diagram of 3D springl. Gray squiggles indicate springs.

Relaxation seeks to find a set of spring lengths $l_{n,m}$ and pose $\vec{b}_n$ that place the system in equilibrium. Equilibrium occurs when the gradient of potential energy

$$VE = \sum_n \sum_m \sum_k \frac{1}{2}\kappa_{\text{ext}} \|q_{n,m} - q_{n,m,k}\|^2 + \sum_n \sum_m \frac{1}{2}\kappa_{\text{int}}(l_{n,m} - 2r)^2 \quad (6)$$

is zero. Global optimization of VE is intractable because external constraints do not have a simple analytic formulation. To make optimization of VE tractable, we consider how to optimize the local potential energy

$$VE(\boldsymbol{S}_n) = \sum_m \sum_k \frac{1}{2}\kappa_{\text{ext}} \|q_{n,m} - q_{n,m,k}\|^2 + \sum_m \frac{1}{2}\kappa_{\text{int}}(l_{n,m} - 2r)^2 \quad (7)$$

assuming that the $q_{n,m,k}$s are fixed.

Solving for the partial derivative of $VE(S_n)$ w.r.t. $l_{n,m}$ and $\vec{b}_n$, we obtain (8) and (9) (see Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.162, for derivation)

$$\frac{\partial VE}{\partial l_{n,m}} = \left(\boldsymbol{M}(\vec{b}_n)\vec{s}_{n,m}\right) \cdot \left(\vec{f}^{\text{int}}_{n,m} + \vec{f}^{\text{ext}}_{n,m}\right) \quad (8)$$

$$\frac{\partial VE}{\partial \vec{b}_n} = \sum_m l_{n,m}\vec{s}_{n,m} \times \vec{f}^{\text{ext}}_{n,m}. \quad (9)$$

The gradient is used to minimize $VE(S_n)$ via the following iterative scheme:

$$l_{n,m}(z+1) = l_{n,m}(z) - \xi \frac{\partial VE}{\partial l_{n,m}}(z), \quad (10)$$

and

$$\vec{b}_n(z+1) = \vec{b}_n(z) - \varrho \frac{\partial VE}{\partial \vec{b}_n}(z), \quad (11)$$

where ξ and ϱ are appropriate step sizes. The step sizes multiply the spring constants κ_{ext} and κ_{int} in (8) and (9), so we decided to fix the step sizes at $\xi = \varrho = 0.1$ and only tune the free parameters κ_{ext} and κ_{int} in later experiments. In an attempt to minimize the global energy (VE), $q_{n,m,k}$s are recomputed after each iteration of (10) and (11) under the constraint that points can only slide along the triangle edge with which they are currently associated. This was done to avoid expensive recomputation of neighborhoods during relaxation.

4.3.2 Nonphysical System

There are other suitable definitions of attraction forces, for which we generalize $\vec{f}^{\text{ext}}_{n,m}$ to use an arbitrary weight $\alpha(x)$ and threshold $\beta(x)$ function. The weighted external force is given by

$$\vec{c}_{n,m} = \sum_k \alpha(\|q_{n,m} - q_{n,m,k}\|)(q_{n,m} - q_{n,m,k}), \quad (12)$$

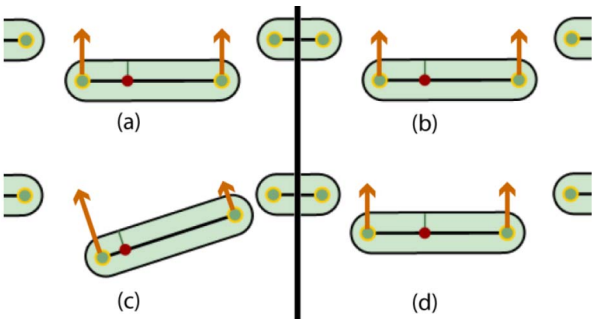


Fig. 6. (a,b) Initial springl configuration and equilibrium configuration with (c) physical torque $\tau_{n,m}$ and (d) normalized torque $\tau^*_{n,m}$. Arrows indicate perpendicular component of $\vec{g}_{n,m}$. The springl does not rotate in (d) because $\vec{\gamma}^*_n(0) = 0$ and the perpendicular component of $\vec{g}_{n,m}$ is constant.

and the thresholded external force is given by

$$\vec{g}_{n,m} = \beta\left(\|\vec{c}_{n,m}\|\right)\vec{c}_{n,m}. \quad (13)$$

The local potential energy equation has been modified to reflect this divergence from a physical system:

$$VE^*(\boldsymbol{S}_n) = \sum_m \frac{1}{2}\|\vec{g}_{n,m}\|^2 + \sum_m \frac{1}{2}\kappa_{\text{int}}(l_{n,m} - 2r)^2. \quad (14)$$

The gradient of $VE^*(\boldsymbol{S}_n)$ is given by

$$\frac{\partial VE^*}{\partial l_{n,m}} = \left(\boldsymbol{M}(\vec{b}_n)\vec{s}_{n,m}\right) \cdot \left(\vec{f}^{\text{int}}_{n,m} + \mathbf{J}^{\text{T}}_{n,m}\vec{g}_{n,m}\right), \quad (15)$$

and

$$\frac{\partial VE^*}{\partial \vec{b}_n} = \sum_m l_{n,m}\vec{s}_{n,m} \times \left(\mathbf{J}^{\text{T}}_{n,m}\vec{g}_{n,m}\right), \quad (16)$$

where $\mathbf{J}_{n,m}$ is the 3×3 Jacobian of $\vec{g}_{n,m}$ w.r.t. $q_{n,m}$. Depending on choices for $\alpha(x)$ and $\beta(x)$, $\mathbf{J}_{n,m}$ could be very difficult to compute. Assuming $\alpha(x)$ and $\beta(x)$ were chosen so that $\mathbf{J}_{n,m}$ is close to an identity matrix, we can use the following iterative scheme that is approximately gradient descent:

$$l_{n,m}(z+1) = l_{n,m}(z) - \xi\left(\boldsymbol{M}\left(\vec{b}_n(z)\right)\vec{s}_{n,m}\right) \left(\vec{f}^{\text{int}}_{n,m}(z) + \vec{g}_{n,m}(z)\right), \quad (17)$$

and

$$\vec{b}_n(z+1) = \vec{b}_n(z) - \varrho\vec{\gamma}_n(z), \quad (18)$$

where

$$\vec{\gamma}_n(z) = \sum_m l_{n,m}(z)\vec{s}_{n,m} \times \vec{g}_{n,m}(z). \quad (19)$$

Selection of $\alpha(x)$ and $\beta(x)$ will be discussed in Section 5.1 on parameter choices.

The purpose of relaxation is to smooth and regularize the constellation while insuring the union of all springl capsules covers the zero iso-level of the signed level set. To this end, we decided not to use physical torque. Fig. 6 illustrates that if we use physical torque $\tau_{n,m} = l_{n,m}\vec{s}_{n,m} \times \vec{g}_{n,m}$ as in a spring-mass system, the equilibrium configuration has increased gap size and disparity in normals and spring lengths. Normalizing torque $\tau^*_{n,m} = \vec{s}_{n,m} \times \vec{g}_{n,m}$ reduces gap size and

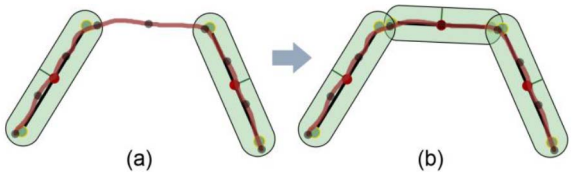


Fig. 7. (a) Exposed zero crossing is detected and (b) covered by inserting a springl.

improves the uniformity of spring lengths. Consequently, $\vec{\gamma}_n$ has been changed to reflect this modification:

$$\vec{\gamma}_n^*(z) = \sum_m \vec{s}_{n,m} \times \vec{g}_{n,m}(z). \tag{20}$$

4.4 Filling Gaps

If springls are unable to cover the zero iso-level through relaxation, gaps must be filled by adding more springls in a subsequent step. When a zero-crossing of the level set is exposed, a springl is added to cover the zero-crossing (**FillGaps**) (Fig. 7). Zero-crossings are evaluated at the centroids of triangles generated from the signed level set's iso-surface via Marching Cubes [33]. These triangles are reused to fill the hole with a springl in the same shape and position. In doing so, the maximum gap between springls cannot not exceed a sphere of radius d_{max}. Assignment of point correspondences for new springls will be discussed in Section 4.7.

4.5 Level Set Evolution

The signed level set provides a redundant, implicit representation of the model, which is parameterized by a constellation of springls. To enforce consistency between the constellation and signed level set, the level set is evolved (**Evolve**) to track the moving constellation. This is done by constructing an unsigned level set $\omega(\boldsymbol{X})$ (23), which is the clamped minimum distance to all springls. The signed level set $\varphi(\boldsymbol{X})$ is evolved to minimize the energy function in (18) (See [2], [3] for a detailed description of level set evolution). The free parameter λ controls the model's smoothness

$$\omega(\boldsymbol{X}) = \min\{d_{max}, d_1(\boldsymbol{X}) \ldots d_N(\boldsymbol{X})\}, \tag{21}$$

and

$$E = \left(\frac{1}{2}\omega(\boldsymbol{X})^2 + \lambda|\nabla\varphi(\boldsymbol{X})|\right)\delta(\varphi(\boldsymbol{X}))d\boldsymbol{X}. \tag{22}$$

The energy function is minimized with the following iterative scheme where $\delta_\varepsilon(x)$ is a compactly supported approximation to the dirac delta.

$$\varphi^{z+1}(\boldsymbol{X}) = \varphi^z(\boldsymbol{X}) - \triangle t\delta_\varepsilon(\varphi^z(\boldsymbol{X}))(\omega(\boldsymbol{X})\nabla\omega(\boldsymbol{X})\cdot \frac{\nabla\varphi^z(\boldsymbol{X})}{|\nabla\varphi^z(\boldsymbol{X})|} + \lambda\nabla\cdot\frac{\nabla\varphi^z(X)}{|\nabla\varphi^z(X)|}). \tag{23}$$

Level sets can change topology when a portion of the model thins below the resolution of the grid. This is sometimes undesirable if the object has a known, fixed topology. Instead of increasing grid resolution in thin regions [23], the level set method can be augmented to explicitly preserve topology [34]. The idea is to prevent a voxel's level set value from changing when a topology change is about to occur. However, springls will continue to move, possibly far from the zero iso-level. The advection equations have been modified to stop springl movement when the topology constraint prevents the zero iso-level from moving

$$\widetilde{\frac{\partial \mathrm{p_n}}{\partial t}}(\mathrm{t}) = \begin{cases} \frac{\partial p_n}{\partial t}(\mathrm{t}), & (\varphi(p_n(t)) - \varphi(p_n(t-1)))(\nabla\varphi(p_n(t)) \cdot \vec{\boldsymbol{n}}_n(t)) \geq 0 \\ 0, & otherwise \end{cases} \tag{24}$$

$$\widetilde{\frac{\partial q_{n,m}}{\partial t}}(\mathrm{t}) = \begin{cases} \frac{\partial q_{n,m}}{\partial t}(\mathrm{t}), & (\varphi(p_n(t)) - \varphi(p_n(t-1)))(\nabla\varphi(p_n(t)) \cdot \vec{\boldsymbol{n}}_n(t)) \geq 0 \\ 0, & otherwise. \end{cases} \tag{25}$$

For large elastic deformations where the CFL number exceeds 1, it is usually faster to convert the unsigned level set to a signed level set (**UnsignedToSigned**) than to evolve the signed level set with active contour methods. This problem can be formulated as watertight surface reconstruction from point clouds, for which there has been extensive research [35], [36], [37], [38], [39], [40].

We perform the unsigned-to-signed conversion quickly by growing the background region (outside the object) and negating the unsigned level set in the foreground region (complement of background). This is done robustly with a coarse-to-fine strategy to prevent the background region from leaking through gaps between springls. To do so, the background is first grown for 16 iterations on a coarse grid ($16 \times 16 \times 16$), starting from the boundary of the volume. The grid resolution is then doubled, and the background is grown for two iterations. The process is repeated until the grid resolution matches the resolution of the level set image. This technique provides a fast coarse estimate of the object's boundary and is robust to gaps between springls. After initial unsigned-to-signed conversion, **Evolve** is used to accurately minimize (24).

4.6 Contraction and Resampling

Springls are resampled to regularize the sampling distribution and triangle quality. Triangles are split along their longest edge if the length of that edge exceeds a threshold (1.5 voxels) (**Resample**). If a triangle's angles fall outside a tolerable range $[20°, 160°]$, then the springl is removed (**Contract**). Removing these springls makes the constellation a more compact representation of the object because these springls are less effective at covering the zero iso-level of the signed level set due to their small surface areas. A springl is destroyed if after evolving the signed level set a springl's particle is more than $(1+\varepsilon)d_{max}$ from the zero iso-level of the signed level set (**Contract**). We choose $\varepsilon = 0.25$ in all cases.

4.7 Tracking

SpringLS maintain a mapping from each particle to the centroid of a triangle on the original model. The initial mapping is an identity mapping ($a_n = p_n$). When a springl is split, the mapping is duplicated and when a springl is added, the mapping is chosen to be either the average point mapping for neighboring springls within a radius of R or the point mapping for the closest springl within a radius of R. If the distance between the average and nearest point correspondence is larger than a threshold (2 voxels), then the nearest point correspondence is used instead of the average. This makes the method robust to situations where neighboring point correspondences are far from each other on the original object, resulting in an average mapping that is far from the object's boundary. Still, this method can produce mappings that lie slightly off the original surface. To prevent correspondence points from drifting from the original surface, the correspondence point is moved along the gradient of the distance (φ_{ref}) to the original surface:

$$a_n(t+1) = a_n(t) - \lambda \varphi_{ref}(a_n(t)) \nabla \varphi_{ref}(a_n(t)), \tag{26}$$

for four iterations ($\lambda = 0.5$). After which, a_n can be projected onto the closest triangle on the original surface and referenced with barycentric coordinates.

4.8 Implementation

Since SpringLS requires updating both a particle system and level set, it has the computational overhead of both. Fortunately, there are solutions for parallelizing particle systems and level sets that enable them to run at interactive frame rates [41], [42], [43], [44]. SpringLS reuses these parallel design patterns and is implemented in OpenCL, a programming language intended for multicore architectures, such as the GPU. Modern CPUs also support OpenCL. We prefer to run OpenCL on the CPU because the CPU can access more global memory, which allows the software to work with larger problem sizes on inexpensive hardware. It also frees up the GPU to be used for real-time volumetric rendering.

For efficient data manipulation on OpenCL devices, a springl is stored as a 4×4 matrix (27), where each row corresponds to a float4 vector.

$$\boldsymbol{S}_n = \begin{bmatrix} p_n^x & p_n^y & p_n^z & \varphi_n \\ q_{n,1}^x & q_{n,1}^y & q_{n,1}^z & a_n^x \\ q_{n,2}^x & q_{n,2}^y & q_{n,2}^z & a_n^y \\ q_{n,3}^x & q_{n,3}^y & q_{n,3}^z & a_n^z \end{bmatrix}. \tag{27}$$

The signed level set value (φ_n) is a temporary variable used to indicate the destruction of a springl. A spatial lookup table is constructed using the uniform grid hashing procedure described in [45]; but instead of spherical particles, springl capsules are used. We prefer the implementation that uses atomics, which has $O(N/P)$ time complexity for N springls and P processing units. It is assumed that a grid cell will not contain more than 16 springls, and a springl capsule will not fall into more than 32 grid cells. The relaxation and resampling phases reinforce these assumptions, but local failures may occur when these assumptions are violated. The unsigned level set $\omega(\boldsymbol{X})$ is computed in a reduction phase that finds the minimum distance among all springl capsules that fall into a grid cell.

Springl neighbors are computed by first enumerating all springls in all hash bins that fall within a distance of R from each vertex $q_{n,m}$. The list is then sorted by springl id to identify duplicates, and at most eight springls within a distance R are considered in computing $\vec{g}_{n,m}$. Nearest neighbors are stored as tuples containing the springl id and edge id for the edge with the closest point $q_{n,m,k}$. During relaxation, the closest point can slide along an edge; but to avoid recomputing neighbors, it is assumed the closest point does not switch to a different edge or an edge on a different triangle.

The advection phase applies a displacement to each springl (see (1) and (2)). The maximum displacement of any springl particle cannot exceed d_{max} because then the zero iso-level of the signed distance function could fall outside the capture range of the unsigned level set. This limitation can be circumvented in one of two ways. Either the unsigned distance field can be extended with fast-marching [46], or the unsigned level set can be converted to a signed level set (**`UnsignedToSigned`**). We do neither in the active contour deformation algorithm and instead choose a small time step ($\triangle t \leq d_{max}/f_{max}(t)$) where $f_{max}(t) = \max_n \| \frac{\partial p_n}{\partial t}(t) \|$.

The relaxation phase computes new springl configurations based on the current constellation and then applies the updates in a second pass to generate springl configurations for the next iteration. The process is repeated for five iterations (see **Algorithm 1.**)

Algorithm 1. **`Relax`**

```
foreach springl n do
  foreach vertex m do
    foreach neighbor k of vertex m do
      Compute α(q_{n,m}, q_{n,m,k})
      Accumulate c⃗_{n,m}
    Compute g⃗_{n,m} and s⃗_{n,m}
    Accumulate γ⃗*_n
  foreach vertex m do
      Compute q^{t+1}_{n,m}
```

SpringLS uses an implementation of the sparse-field level set method [47] that has been parallelized. After evolving the level set with (23) (**`Evolve`**), the signed distance field is rebuilt (**`Rebuild`**) in the region near the model's boundary, commonly referred to as the narrow band. Indexes into the narrow band are stored in a buffer to avoid traversal of the entire level set volume. The narrow-band buffer is updated after each pass of **`Rebuild`**. Since parallel narrow-band implementations are nontrivial, we refer the reader to [42], [44]. The time complexity of the parallel sparse-field algorithm is $O(M^2/P)$ for M^3 voxels and P processing units.

Algorithm 2 describes the complete springls deformation algorithm for K iterations and a resampling period of M (i.e., the contraction and resampling phases are performed every Mth iteration). We choose $M = 5$ in all cases. This algorithm is intended for fluid-like deformations, but there are other acceptable variations on the SpringLS deformation algorithm. For instance, **Algorithm 3** is a more elastic version of the deformation algorithm that permits

large step sizes; or alternatively, the model behaves like a mesh (without triangle connectivity) if only **Advect** is used. If **Contract**, **Resample**, and **FillGaps** are disabled, a 1-to-1 mapping between springls is maintained. If **Relax** is disabled for some springls, then corresponding geometric features on the initial mesh are preserved.

Algorithm 2. **Active Contour Deformation**

```
for k = 1 : K do
  Advect
  Relax
  if k mod M == 0 then
    Contract
    Resample
    Relax
    for z = 1 : 4 do
      Evolve
      Rebuild
    FillGaps
  else
    for z = 1 : 4 do
      Evolve
      Rebuild
```

Algorithm 3 has no CFL time step restriction because it uses **UnsignedToSigned** to reconstruct the signed level set at each time step. Depending on the step size, **UnsignedToSigned** may be more efficient than level set evolution. The choice between algorithms depends on the problem and if there is already a step size restriction due to the nature of the problem (e.g., fluid simulations).

Algorithm 3. Elastic Deformation

```
for k = 1 : K do
  Advect
  UnsignedToSigned
  for z = 1 : 4 do
    Evolve
    Rebuild
  Relax
  FillGaps
  Contract
  Resample
```

5 Results

5.1 Parameter Choices

There are many parameters associated with the SpringLS representation, some intrinsic to the model and others image/task dependent. When selecting intrinsic parameters, the primary concern is to choose those that hold the constellation of springls together during deformation as much as possible. The **FillGaps** step is a catch-all to fix holes in the constellation that occur due to advection and relaxation. However, it is desirable to minimize the amount of gap filling because it requires the introduction of more springls and interpolation of new point correspondences. Consider a pathological case where all springls are destroyed in the **Contract** step. In the case where $\lambda = 0$ (i.e., no regularization in (24)), the **Evolve** step will not move the interface because the unsigned level set is d_{max} everywhere. The **FillGaps** step will then regenerate a constellation of springls that consists of all triangles in the iso-surface. Even though the deformable model was prevented from tearing, all tracking information was lost and the model failed to deform. We would like to preserve the constellation with relaxation to avoid this scenario.

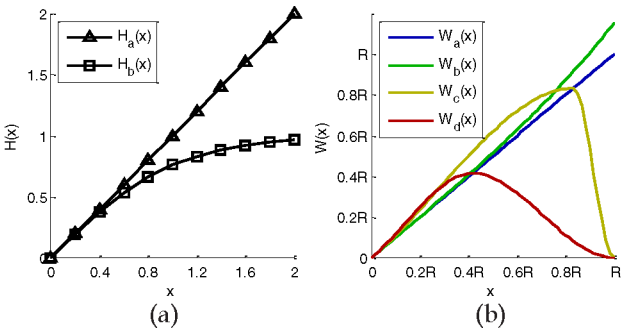


Fig. 8. (a) $\mathrm{H}(x)$ and (b) $\mathrm{W}(x)$ are considered in parameter tuning experiments. These functions are related to threshold $\beta(x)$ and weight $\alpha(x)$ functions via $\beta(x) = \frac{1}{\mathrm{x}}\mathrm{H}(\lambda_c x)$ and $\alpha(x) = \frac{1}{\mathrm{x}}\mathrm{W}(\frac{x-2r}{R})$.

Many parameters can be chosen heuristically (Table 1), leaving the spring constant κ_{int} and functions $\alpha(x)$ and $\beta(x)$ to be chosen empirically. For weight and threshold functions of the form $\alpha(x) = \frac{1}{\mathrm{x}}\mathrm{W}(\frac{x-2r}{R})$ and $\beta(x) = \frac{1}{\mathrm{x}}\mathrm{H}(\lambda_c x)$, we looked at two choices for $\mathrm{H}(x)$: $\mathrm{H_a}(x) = x$ and $\mathrm{H_b}(x) = \tanh(x)$, and four choices for $\mathrm{W}(x)$: $\mathrm{W_a}(x) = \mathrm{x}$, $\mathrm{W_b}(x) = \mathrm{atanh(x)}$, and $\mathrm{W_c}(x)$ and $\mathrm{W_d}(x)$ are Hermite splines satisfying

$$
\begin{array}{ll}
\mathrm{W_c}(0) = 0 & \mathrm{W_d}(0) = 0 \\
\mathrm{W_c}(d_{max}) = d_{max} & \mathrm{W_d}(d_{max}/2) = d_{max}/2 \\
\mathrm{W_c}(d \geq R) = 0 & \mathrm{W_d}(d \geq R) = 0 \\
\dot{\mathrm{W}}_\mathrm{c}(0) = 1 & \dot{\mathrm{W}}_\mathrm{d}(0) = 1 \\
\dot{\mathrm{W}}_\mathrm{c}(d_{max}) = 0 & \dot{\mathrm{W}}_\mathrm{d}(d_{max}/2) = 0 \\
\dot{\mathrm{W}}_\mathrm{c}(R) = 0 & \dot{\mathrm{W}}_\mathrm{d}(R) = 0.
\end{array}
$$

See Fig. 8 for depiction of functions. λ_c is a free parameter that will be referred to as the sharpness of $\mathrm{H}(x)$. A repulsion force appears in $\alpha(x)$ to keep springls at a minimum distance of $2r$. This is analogous to the repulsion force used in particle systems to bound the number of particles that can occupy a grid cell without overlap. The data structure used to index springls assumes that there is a bound on the number of springls that can occupy a grid cell [45]. The repulsion force reinforces this assumption.

$\mathrm{H_a}(x)$ paired with $\mathrm{W_a}(x)$ causes the attractive forces between springls to behave like springs. A potential drawback of these functions is that an imbalance in the number of neighboring springls will cause an imbalance in the rotational forces exerted on a springl, even if the proximity of neighboring springls is the same. This is due to the sum that appears in the expression for the external force (12). To address this concern, we propose $\mathrm{H_b}(x)$ paired with $\mathrm{W_b}(x)$ to dampen the cumulative force that can be exerted by neighboring springls. However, both sets of functions have infinite support even though the force is only applicable to springls within a distance of R. To assess whether this is a problem, we propose $\mathrm{W_c}(x)$ and $\mathrm{W_d}(x)$; for which, the force exerted by neighboring springls smoothly decreases to zero within its neighbor range.

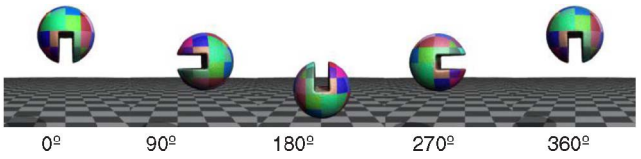


Fig. 9. Zalesak's sphere advected with a rotational velocity field. The initial (0 degree) and final (360 degree) constellation are almost identical. Grid dimensions were $256 \times 256 \times 256$.

5.2 Parameter Tuning

Enright et al. [48] introduced two tasks that test extremes of deformable model behavior. The first test places a notched sphere (Zalesak's sphere) in a velocity field that rotates the object 360 degree every 628 time steps (Fig. 9). The second test places a sphere in an incompressible flow field proposed by LeVeque [49] that stretches the object into a disk. The flow is time-reversed by modulating the velocity field with $s(t) = \cos(\pi t/T)$. The model achieves maximal deformation at time $t = T/2$ and returns to a sphere at $t = T$, where T controls the maximum amount of deformation (Fig. 10). In both tests (Figs. 9 and 10), particle advection was integrated in time with a fourth order Runge-Kutta scheme.

The level set method performs poorly on both tests by failing to preserve shape and volume, which is important in simulation of solids and fluids. LeVeque's test is difficult for mesh representations because it requires remeshing. There have been several papers that present deformable model representations which perform better than the level set method on LeVeque's test. Wicke et al. [12] use a tetrahedral mesh, Bargteil et al. [15] use semi-Lagrangian contouring, and several other methods [12], [15], [24], [28], [50] use a level set coupled with a set of particles. SpringLS parameters were selected based on LeVeque's test because it has become a popular benchmark.

For the purposes of parameter tuning, LeVeque's test was evaluated on a small grid (128^3) for a maximum deformation

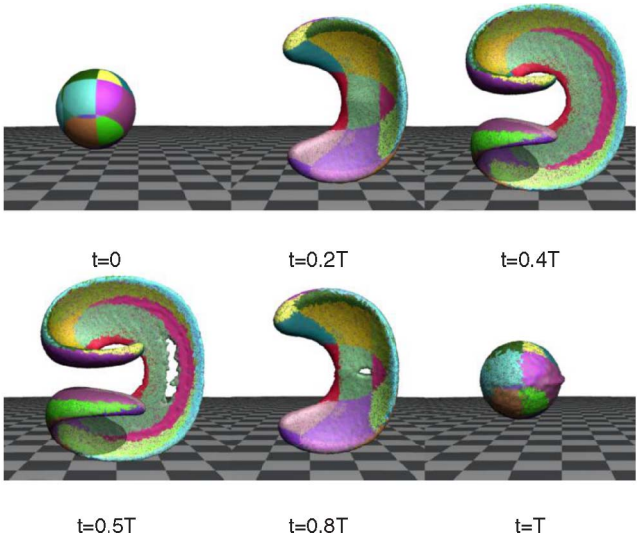


Fig. 10. LeVeque's test showing sphere deformed into a disk. The model reaches its maximal deformation at $t = 0.5\{\rm T\}$ and returns to a sphere at time $t = \mathrm{T}$ for a period of $\mathrm{T} = 3$. Grid dimensions were $256 \times 256 \times 256$. Simulation used 12K-32K springls.

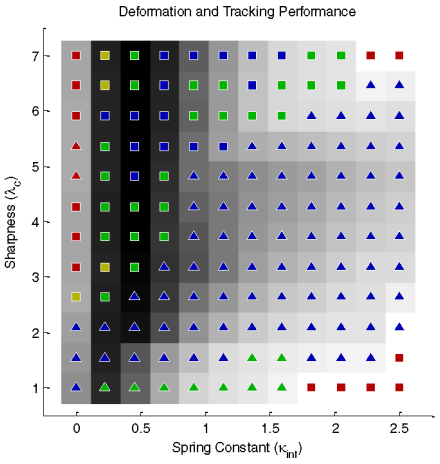


Fig. 11. Plot of the minimal total error (deformation plus tracking error) among all pairs of threshold/weight functions. Darker regions have lower error. Glyphs indicate the best performing set of threshold/weight functions. The shape and color of each glyph corresponds to the functions in Fig. 8.

of $T = 2.5$. We looked at four different metrics: deformation error, tracking error, average number of springls, and total number of added and removed springls (Fig. 11).

Deformation error refers to the point surface distance between particle locations p_n and the original model $\varphi_{ref}(X)$, or simply $|\varphi_{ref}(p_n)|$. Tracking error refers to the distance between a_n and p_n for each springl, or $\|p_n - a_n\|$. For each set of parameters, all eight combinations of weight and threshold functions were evaluated. Fig. 11 shows the minimum total error (deformation plus tracking error) among all pairs of threshold/weight functions. $\mathrm{H_b(x)}$ paired with either $\mathrm{W_a(x)}$ or $\mathrm{W_b(x)}$ perform the best. We chose to use $\mathrm{H_b(x)}$ and $\mathrm{W_a(x)}$ because $\mathrm{W_a(x)}$ is faster to evaluate than $\mathrm{W_b(x)}$. For this choice of functions, there is a sizable subset of parameters that achieve similar performance (Fig. 12b). Subsequent experiments use $\kappa_{int} = 0.3$ and $\lambda_c = 5$, and *none* of the intrinsic parameters are changed or retuned.

5.3 Computational Performance

Performance was evaluated on a PC with Dual Intel E5630s. Parameter choices reflect the optimal ones selected in the previous section. LeVeque's test was repeated on different size grids for a period of $T = 2.5$. Table 3 summarizes these experiments. Deformation and tracking errors are almost

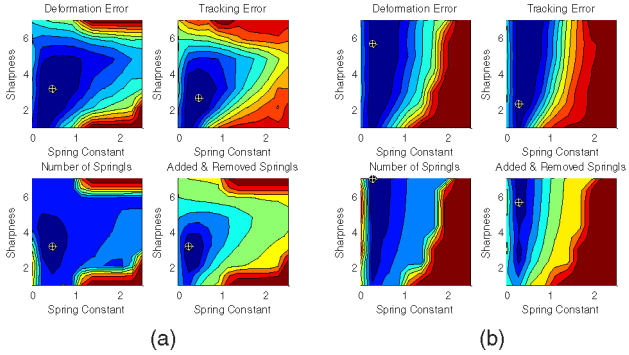


Fig. 12. Plot of performance metrics for (a) $\mathrm{H_a(x)}$ and $\mathrm{W_a(x)}$ and (b) $\mathrm{H_b(x)}$ and $W_a(x)$. The minimum value is marked with a "$\oplus$."

TABLE 3
Performance on LeVeque's Test ($T = 2.5$)

Grid Size	Mean # Springls	Deformation Error (voxels)	Tracking Error (voxels)	Total Iterations	Total Time
128^3	5K	0.38±0.28	0.93±1.07	500	35 sec
256^3	19K	0.34±0.31	0.92±1.05	1000	300 sec
384^3	41K	0.31±0.27	0.91±1.08	1500	970 sec
512^3	72K	0.32±0.30	0.94±1.18	2000	2500 sec

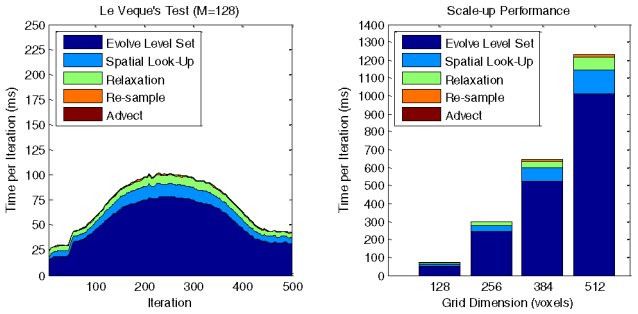


Fig. 13. Computation time per iteration as a function of (a) iteration and (b) grid dimension for LeVeque's test.

TABLE 4
Performance on Zalesak's Test for Grid Size 128^3

Method	Re-sampling	Device	Speed	Volume Change	Mean # Particles
SpringLS	Yes	CPU	20 fps	0.6%	2.8K
PLS	Yes	CPU	2.5 fps	2.6%	80K
PLS	No	CPU	3 fps	0.3%	86K
MarkerLS	No	GPU	28 fps	0.9%	49K

The Particle Level Set library (PLS) [51] uses 1 core. SpringLS uses 8 cores. MarkerLS from Mei et al. [50] does not re-sample.

constant *in voxel units*. Fig. 13 shows the computational performance as a function of total iterations and grid size. The algorithm spends between 76 percent (128^3 grid) to 82 percent (512^3 grid) of the time in level set evolution. This time can be reduced by adopting a more work efficient level set method that runs on the GPU [42].

We also repeated Zalesak's test at a grid size of 128^3 to compared it against implementations of Particle Level Sets (PLS) [51] and Marker Level Sets [50] (Table 4). Finally, we compared SpringLS to MarkerLS for LeVeque's test ($T = 2.5$ and grid size 128^3) based on published results. The volume change for MarkerLS was 1.8 percent and SpringLS was 4.6 percent. However, the MarkerLS implementation [50] does not resample and uses 98K particles for LeVeque's test. This is compared to SpringLS that uses only 5K springls on average and does re-sample. If we increase the grid size to 256^3 for SpringLS, 19K springls are used on average and the volume change decreases to 1.2 percent.

5.4 Tracking Performance

Tracking performance is important for imaging applications, but there has been little quantitative reporting of it in literature for remeshing and resampling schemes. Fig. 14 shows SpringLS tracking and deformation performance on Enright's tests at a grid resolution of $256 \times 256 \times 256$. Deformation and tracking errors were measured for

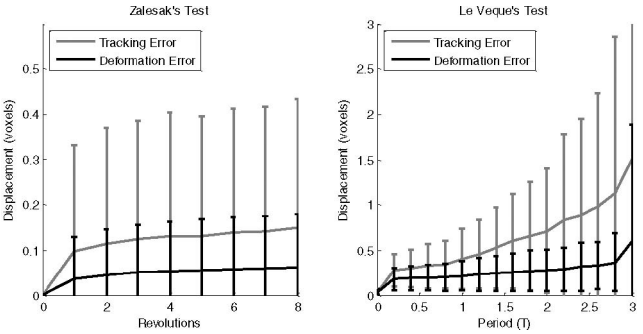


Fig. 14. Deformation and tracking errors for Zalesak's and LeVeque's test for grid size 128^3. Distances are measured in voxels.

TABLE 5
Comparison with Other Methods for LeVeque's Test ($T = 1.6$)

Method	Grid Size	Mean Deformation Error (voxels)	Mean Tracking Error (voxels)
LSID	100^3	0.34	1.7
LSPC	100^3	0.34	0.63
SpringLS	**100^3**	**0.26**	**0.57**
LSID	200^3	0.38	1.4
LSPC	200^3	0.38	0.68
SpringLS	**200^3**	**0.25**	**0.59**

Comparison with published results [52] for Level Sets with Interfacial Data (LSID) and Level Sets with a Point Correspondence (LSPC).

Zalesak's test at the end of each revolution, and they were measured for LeVeque's test after a full deformation period. Errors for Zalesak's test are subvoxel, and errors for LeVeque's test increase linearly with the amount of deformation up until $T = 3$; at which point, the object tears during deformation and error increases significantly.

Finally, we compared SpringLS with implicit level set tracking [52]. SpringLS performs slight better (Table 5), but the main advantage, which we will demonstrate later, is that SpringLS can circumvent the CFL time step restriction for methods that require large step sizes.

One limitation of SpringLS is that the density of springls is tied to the resolution of the level set through Marching Cubes in the **`FillGaps`** step, whereas representations that use particles can always increase the density of particle sampling to improve accuracy, regardless of grid resolution. LeVeque's test used 5K springls on average at a grid size of 128^3 whereas the Marker Level Set method used 98K particles at a grid size of 128^3 [50]. More particles translates into more degrees of freedom, which implies lower material viscosity. For this reason, SpringLS are not as good at modeling materials with low viscosity or for problems that require accuracy well below the grid resolution. This is less a concern for imaging applications where image resolution, contrast, and noise impose limits on segmentation accuracy.

5.5 Topology-Preserving Deformations

Spring Level Sets can rigorously preserve topology by using the topology preserving level set method [34]. To demonstrate this ability, LeVeque's test was repeated with the topology-preservation constraint. The object's topology was preserved in Fig. 15 under extreme deformations even though the constellation has no explicit connectivity.

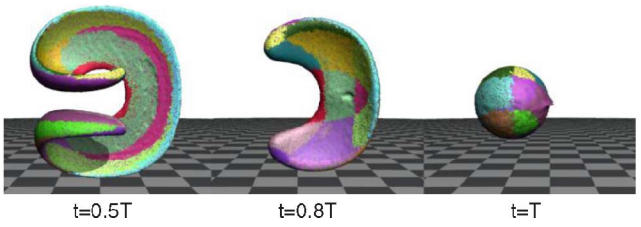


Fig. 15. LeVeque's test with topology-preservation constraint for period $T = 3$.

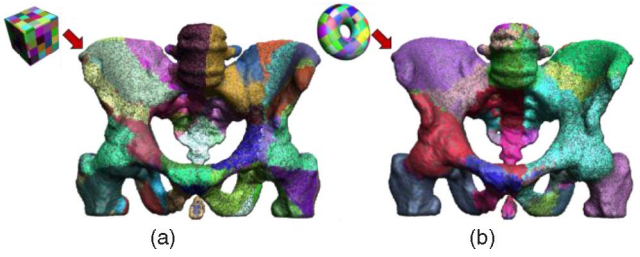


Fig. 16. Nonatlas-based pelvis segmentation when initialized with a (a) cube and (b) torus.

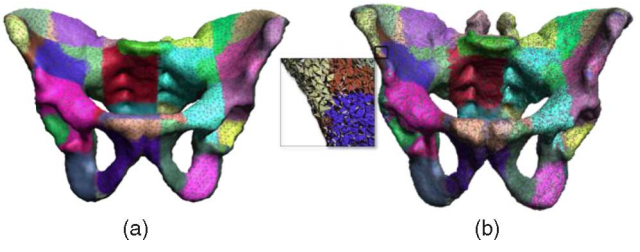


Fig. 17. (a) Mean atlas shape and (b) segment pelvis after active contour segmentation with SpringLS when initialized with PDM.

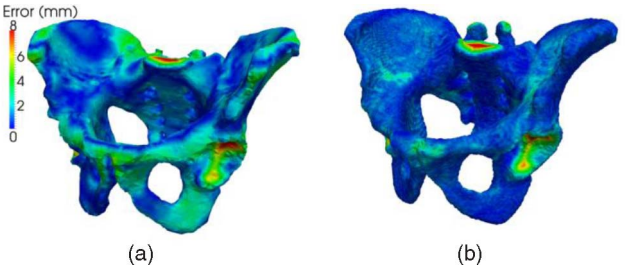


Fig. 18. Atlas-based segmentation showing (a) registered PDM and (b) final active contour segmentation with SpringLS.

TABLE 6
Summary of Image Segmentation Results

Experiment	Iterations	Mean # Springls	Time
Pelvis	400	49K	220-240 sec
Pelvis \w atlas	100	25K	30 sec

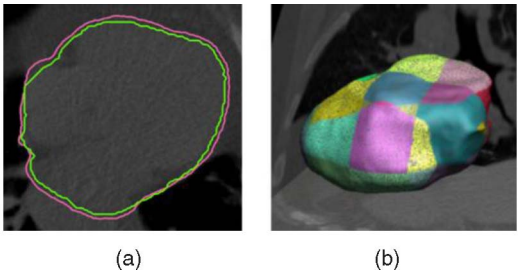


Fig. 19. (a) Manual segmentation of one time frame shown in green and active contour segmentation shown in pink. After active contour segmentation, the epicardium is tracked to all other time frames. (b) shows one time frame of tracked epicardium.

5.6 Image Segmentation

Spring Level Sets were applied to active contour image segmentation [53] of objects driven under pressure forces from image intensities. It was implemented as a mixture of Java and OpenCL for the CPU. Parameter settings and grid size 256^3 were fixed for all experiments. Experiments not initialized with an atlas were repeated with four different initializations (sphere, cube, torus, and meta-sphere [7]). Reported segmentation errors are measured as the average minimum distance from mesh vertices on the segmented mesh to iso-surface produced with standard level set segmentation.

Fig. 16 shows segmentation of a pelvis from a CT image and highlights SpringLS's ability to change topology. Initializing the segmentation process with a cube or torus causes segmentation of the pelvis to include the femurs and spine as well. To mitigate this problem, we incorporate an atlas-based approach.

The segmentation result in Fig. 16 can be improved by combining level set techniques with a parametric atlas. A Point Distribution Model of the pelvis was constructed with the standard PCA method [6] (Fig. 17). Analogous statistical atlas methods have been developed for level sets [54], but these representations are not equivalent. The atlas was registered (rigid + global scale) to the CT image. The first 10 mode weights were optimized in increasing order to reduce the average distance from the atlas to target iso-surface in CT. Because the initial registration was fairly good $(1.56 \pm 1.36\ \mathrm{mm})$, optimizing the mode weights modestly improved the segmentation result to $1.40 \pm 1.32\ \mathrm{mm}$. The registered mesh was then treated as a constellation of springls and advected toward the target iso-level with an external velocity field produced by Gradient Vector Flow (GVF) [55] and pressure forces. This atlas-based method produces a better pelvis segmentation (Figs. 17 and 18) than the nonatlas approach and provides a mapping from each springl back to the atlas, enabling transfer of region labels on the atlas to the segmented pelvis. This technique is also significantly faster than the nonatlas-based approach. Table 6 summarizes results from these experiments. Segmentation results are very similar to performing the same task with the Level Set Method. The average distance between the SpringLS and LSM segmentations for the nonatlas approach was $0.13 \pm 0.15\ \mathrm{mm}$ and atlas approach was $0.25 \pm 0.32\ \mathrm{mm}$.

5.7 Tracking in 4D CT

To demonstrate elastic and fluid behaviors of Spring Level Sets, we applied SpringLS to segmentation and tracking of objects in 4D CT (a time series of 3D CT images). The following imaging pipeline combines segmentation, registration, and atlasing into a single system that uses SpringLS as its only deformable model representation. In lieu of a statistical atlas, or to construct one, we first manually segment objects of interest in one CT image. In this case, the epicardium. The coarse manual segmentation is then improved with active contour segmentation methods (Fig. 19a). CT images at all other times are nonrigidly registered [56] to the reference time frame. The registration

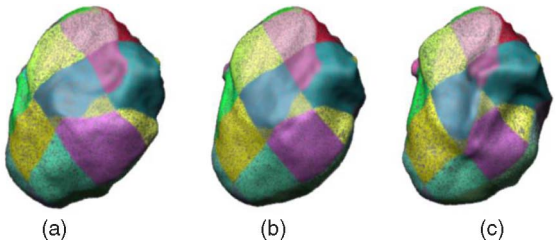


Fig. 20. Iso-surface for heart PDM showing the first mode at (a) -2σ from the mean shape, (b) mean shape, and (c) $+2\sigma$ from the mean shape.

produces a displacement vector field representing an elastic deformation of the source image into the reference image. To track objects, the displacement field is applied to springls followed by **`UnsignedToSigned`**, **`Evolve`**, and **`Rebuild`** (Fig. 19b). The result is a 1-to-1 mapping of springls found in the reference time frame to all other time frames. Because the mapping is 1-to-1, the rigid component of motion can be factored out and the nonrigid component analyzed with PCA to construct the statistical atlas shown in Fig. 20.

The 4D tracking pipeline has only one transition between level set and mesh techniques (Fig. 19). To further motivate the need for Spring Level Sets, we apply them to a problem that requires repeated transitioning between level set and mesh techniques.

5.8 Morphing Animated Characters

There is commercial interest in animated characters that undergo large nonrigid deformations (morph) as they execute a kinematic motion [57]. Morphing is useful for reconstruction of animated characters as well [58]. To address this class of problems, we present the following pipeline for morphing animated characters with SpringLS.

The pipeline begins with two mesh models (source and target) and a kinematic animation acquired with motion capture (Fig. 21). The models are first autorigged with Pinnochio [59] to fit a kinematic skeleton and assign bone weights to the mesh. The source model is then nonrigidly rescaled to align its skeleton with the target model, and both models are converted into SpringLS. During each animation time step, both the source and target model are articulated with Linear Blend Skinning (LBS) [60] to match kinematics of the animation.

Linear Blend Skinning is a type of elastic deformation that is popular in real-time character animation. Associated with each mesh vertex is a vector of bone weights indicating the contribution of each bone to the deformation of the vertex. The springl constellation is deformed by applying a linear combination of the skeleton's rigid bone transforms to triangle vertices, weighted by the bone weights.

Linear Blend Skinning is followed by 10 active contour iterations to deform the source model into the target model incrementally. The active contour deformation is driven by pressure forces derived from the target model's signed level set. The process of alternating between "elastic" LBS deformation and "fluid" active contour deformation is repeated until the source model aligns with the target. Bone weights are tracked from source to target using springls. In order to apply kinematics, it is necessary to track bone weights during LBS deformation.

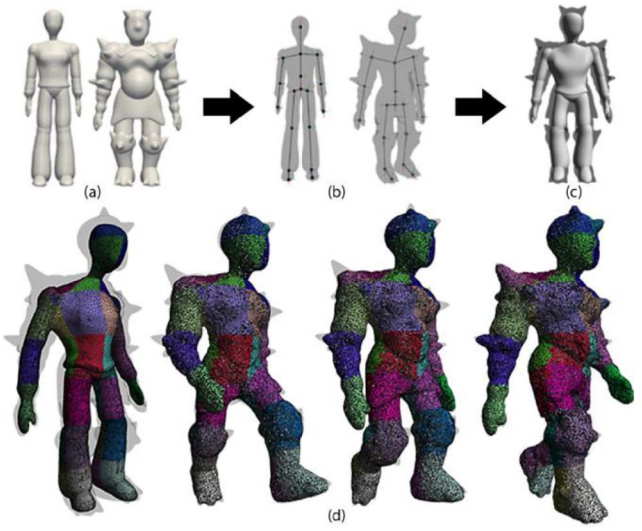


Fig. 21. Morphing pipeline showing (a) input meshes, (b) autorigged skeletons, (c) mesh rescaling to align skeletons, (d) morphing of a walking humanoid character. Springls constellation for morphing character is shown overlaid on silhouette of target character. Meshes courtesy of Cosmic Blobs (Dassault Systems SolidWorks Corp.).

Spring Level Sets guarantee that at all times during the animation, the model is watertight, nonself-intersecting, and in point-to-point correspondence with the original model. This morphing technique is advantageous in that it does not require point correspondences between source and target meshes; and in fact, could be used to establish point correspondences.

6 Discussion

Spring Level Sets are distinguished from previous work by their treatment of meshes as a constellation of disconnected triangles. SpringLS can behave as either a mesh or level set to perform better on tasks which require behaviors of both (Table 7).

Springls are similar to surfels, for which there is work to support the claim that surfels are equally effective at modeling fluid and elastic deformations as other representations [61], [62]. Springls can be transformed into surfels using a springl's particle location to define a disk's center and gradient of the signed level to define the disk's normal (see Fig. 3). However, there are reasons to choose springls

TABLE 7
Comparison of Deformable Model Representations

Model Behavior	Mesh	Level Set
Rigid Transform	✓	
Elastic Deformation	✓	
Topology Preservation	✓	
Point Tracking	✓	
Plastic Deformation		✓
Topology Change		✓
Self-intersection Prevention		✓
Collision Handling		✓

Check marks indicate the preferred representation for each behavior.

over surfels. First, SpringLS can be interpreted as mesh or level set, so they can interact naturally with other geometric models during collision detection and rendering; and second, there is no shape, topology, or tracking information lost in the transformation from mesh or level set into SpringLS. This is crucial for applications that alternate between mesh and level set deformations, as demonstrated by the character morphing example.

Shape preservation is affected by the choice of interspringl forces. Although we have empirically demonstrated one choice of interspringl forces perform well on numerous examples, there are other choices that may have desirable properties. One may also choose not to relax or resample a subset of springls to preserve sharp features and fine details.

SpringLS spends a large percentage of computation time evolving a level set to keep the model's implicit representation consistent with the model's parametric representation. This redundancy and overhead is not unique to SpringLS. Marker/particle level set methods [15], [24], [48] and some re-meshing schemes [8], [10], [15] also maintain an implicit representation to provide a fast inside/outside test and handle topology changes.

7 Limitations

There are limitations to the SpringLS representation. As previously mentioned, the size of surface elements is tied to the Marching Cubes algorithm in the `FillGaps` step. Upsampling the number of surface elements requires upsampling the grid, and the grid's resolution imposes a CFL restriction on step size for active contour deformation (**Algorithm 2**). Fortunately, SpringLS has a method for circumventing the CFL time step restriction (**Algorithm 3**).

SpringLS only tracks points, and by doing so, sacrifices the continuity of the mapping between source and target shapes. Such sacrifice is necessary to address deformations where the object changes topology and the continuity of the mapping is disrupted. Surfels and SpringLS share these challenges, and we refer the reader to the literature on surfels for possible solutions [25].

8 Conclusion

Spring Level Sets merge meshes and level sets into a single representation to provide interoperability between methods designed for either. Because SpringLS uses disconnected surface elements, the object can change topology, track points, and undergo fluid and elastic deformations. The auxiliary level set provides a watertight representation of the model's boundary that cannot self-intersect, and simple rules have been described for adding and destroying surface elements based on the level set representation.

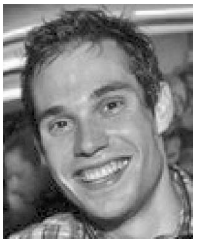
Acknowledgments

This work was supported by a graduate fellowship from the Johns Hopkins Applied Physics Laboratory and internal funds from Johns Hopkins University. The authors thank Dr. Todd McNutt, Dr. Ted Deweese, and Dr. Lee Myers for providing imaging data.

References

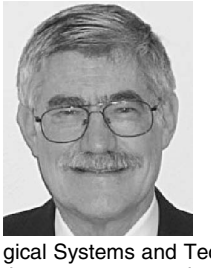
[1] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically Deformable Models," *Computer Graphics,* vol. 21, pp. 205-214, 1987.
[2] J. Sethian, *Level Set Methods and Fast Marching Methods.* Cambridge Univ. Press, 1999.
[3] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, 2003.
[4] M.S.T. Kohlberger, J. Zhang, N. Birkbeck, J. Wetzl, J. Kaftan, J. Decklerck, and S.K. Zhou, "Automatic Multi-Organ Segmentation Using Learning-Based Segmentation and Level Set Optimization," *Proc. Int'l Conf. Medical Image Computing and Computer Assisted Intervention (MICCAI),* 2011.
[5] D. Tosun and J. Prince, "A Geometry-Driven Optical Flow Warping for Spatial Normalization of Cortical Surfaces," *IEEE Trans. Medical Imaging,* vol. 27, no. 12, pp. 1739-1753, Dec. 2008.
[6] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham, "Active Shape Models-Their Training and Application," *Computer Vision and Image Understanding,* vol. 61, pp. 38-59, 1995.
[7] B.C. Lucas, M. Kazhdan, and R.H. Taylor, "SpringLS: A Deformable Model Representation to Provide Interoperability between Meshes and Level Sets," *Proc. Int'l Conf. Medical Image Computing and Computer Assisted Intervention (MICCAI),* 2011.
[8] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming Meshes that Split and Merge," *ACM Trans. Graphics,* vol. 28, article 76, 2009.
[9] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent Advances in Remeshing of Surfaces," *Shape Analysis and Structuring,* pp. 53-82, Springer, 2008.
[10] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Physics-Inspired Topology Changes for Thin Fluid Features," *ACM Trans. Graphics,* vol. 29, article 50, 2010.
[11] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming Meshes that Split and Merge," *Proc. ACM SIGGRAPH,* p. 76, 2009.
[12] M. Wicke, D. Ritchie, B.M. Klingner, S. Burke, J.R. Shewchuk, and J.F. O'Brien, "Dynamic Local Remeshing for Elastoplastic Simulation," *ACM Trans. Graphics,* vol. 29, article 49, 2010.
[13] A. Zaharescu, E. Boyer, and R. Horaud, "Transformesh: A Topology-Adaptive Mesh-Based Approach to Surface Evolution," *Proc. Eighth Asian Conf. Computer Vision (ACCV '07),* pp. 166-175, 2007.
[14] C. Wojtan and G. Turk, "Fast Viscoelastic Behavior with Thin Features," *ACM Trans. Graphics,* vol. 27, pp. 1-8, 2008.
[15] A.W. Bargteil, T.G. Goktekin, J.F. O'brien, and J.A. Strain, "A Semi-Lagrangian Contouring Method for Fluid Simulation," *ACM Trans. Graphics,* vol. 25, pp. 19-38, 2006.
[16] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically Based Deformable Models in Computer Graphics," *Computer Graphics Forum,* vol. 25, pp. 809-836, 2006.
[17] E. Grinspun, P. Krysl, and P. Schröder, "CHARMS: A Simple Framework for Adaptive Simulation," *ACM Trans. Graphics,* vol. 21, pp. 281-290, 2002.
[18] S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross, "Unified Simulation of Elastic Rods, Shells, and Solids," *ACM Trans. Graphics,* vol. 29, pp. 1-10, 2010.
[19] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa, "Point Based Animation of Elastic, Plastic and Melting Objects," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '04),* pp. 141-151, 2004.
[20] D. Gerszewski, H. Bhattacharya, and A.W. Bargteil, "A Point-Based Method for Animating Elastoplastic Solids," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '09),* pp. 133-138, 2009.
[21] A.W. Bargteil, C. Wojtan, J.K. Hodgins, and G. Turk, "A Finite Element Method for Animating Large Viscoplastic Flow," *ACM Trans. Graphics,* vol. 26, article 16, 2007.
[22] M. Müller, "Fast and Robust Tracking of Fluid Surfaces," *Proc. ACM SIGGRAPH,* pp. 237-245, 2009.
[23] D. Enright, F. Losasso, and R. Fedkiw, "A Fast and Accurate Semi-Lagrangian Particle Level Set Method," *Computers and Structures,* vol. 83, pp. 479-490, 2005.
[24] V. Mihalef, D. Metaxas, and M. Sussman, "Textured Liquids Based on the Marker Level Set," *Computer Graphics Forum,* vol. 26, pp. 457-466, 2007.
[25] M. Pauly, R. Keiser, L.P. Kobbelt, and M. Gross, "Shape Modeling with Point-Sampled Geometry," *ACM Trans. Graphics,* vol. 22, pp. 641-650, 2003.

[26] M. Gross, "Point Based Graphics: State of the Art and Recent Advances," *Proc. ACM SIGGRAPH,* 2009.
[27] D. Levin, "Mesh-Independent Surface Interpolation," *Geometric Modeling for Scientific Visualization,* vol. 3, pp. 37-49, 2003.
[28] S.E. Hieber and P. Koumoutsakos, "A Lagrangian Particle Level Set Method," *J. Computational Physics,* vol. 210, pp. 342-367, 2005.
[29] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, "Surfels: Surface Elements as Rendering Primitives," *Proc. ACM SIGGRAPH,* pp. 335-342, 2000.
[30] M.D. Meyer, P. Georgel, and R.T. Whitaker, "Robust Particle Systems for Curvature Dependent Sampling of Implicit Surfaces," *Proc. Conf. Shape Modeling and Applications,* pp. 124-133, 2005.
[31] A.P. Witkin and P.S. Heckbert, "Using Particles to Sample and Control Implicit Surfaces," *Proc. 21st Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 269-277, 1994.
[32] J. Huang, Y. Li, R. Crawfis, S.C. Lu, and S.Y. Liou, "A Complete Distance Field Representation," *Proc. Conf. Visualization,* 2001.
[33] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *ACM SIGGRAPH Computer Graphics,* vol. 21, pp. 163-169, 1987.
[34] X. Han, C. Xu, and J. Prince, "A Topology Preserving Level Set Method for Geometric Deformable Models," *Proc. IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 6, pp. 755-768, June 2003.
[35] P. Mullen, F. De Goes, M. Desbrun, D. Cohen Steiner, and P. Alliez, "Signing the Unsigned: Robust Surface Reconstruction from Raw Pointsets," *Computer Graphics Forum,* vol. 29, pp. 1733-1741, 2010.
[36] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, and D. Cohen-Or, "Competing Fronts for Coarse-to-Fine Surface Reconstruction," *Computer Graphics Forum,* vol. 25, pp. 389-398, 2006.
[37] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," *Proc. Fourth Eurographics Symp. Geometry Processing (SGP '06),* pp. 61-70, 2006.
[38] A. Hornung and L. Kobbelt, "Robust Reconstruction of Watertight 3D Models from Non-Uniformly Sampled Point Clouds without Normal Information," *Proc. Eurographics Symp. Geometry Processing (SGP '06),* pp. 41-50, 2006.
[39] S. Fleishman, D. Cohen-Or, and C.T. Silva, "Robust Moving Least-Squares Fitting with Sharp Features," *ACM Trans. Graphics,* vol. 24, pp. 544-552, 2005.
[40] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," *Proc. ACM SIGGRAPH,* pp. 303-312, 1996.
[41] H. Mostofi, J. Schnabel, and V. Grau, "Fast Level Set Segmentation of Biomedical Images Using Graphics Processing Units," 2009.
[42] M. Roberts, J. Packer, M.C. Sousa, and J.R. Mitchell, "A Work-Efficient GPU Algorithm for Level Set Segmentation," *Proc. Conf. High Performance Graphics (HPG '10),* pp. 123-132, 2010.
[43] K. Crane, I. Llamas, and S. Tariq, "Real-Time Simulation and Rendering of 3D Fluids," *GPU Gems,* vol. 3, pp. 633-675, 2007.
[44] A.E. Lefohn, J.M. Kniss, C.D. Hansen, and R.T. Whitaker, "Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware," *Proc. IEEE Visualization,* p. 11, 2003.
[45] S. Green, "Particle Simulation Using CUDA," White Paper, NVIDIA, Dec. 2010.
[46] J.A. Sethian, "A Fast Marching Level Set Method for Monotonically Advancing Fronts," *Proc. Nat'l Academy of Sciences of the United States of Am.,* vol. 93, pp. 1591-1595, 1996.
[47] R. Whitaker, "A Level-Set Approach to 3D Reconstruction from Range Data," *Int'l J. Computer Vision,* vol. 29, pp. 203-231, 1998.
[48] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A Hybrid Particle Level Set Method for Improved Interface Capturing," *J. Computational Physics,* vol. 183, pp. 83-116, 2002.
[49] R.J. LeVeque, "High-Resolution Conservative Algorithms for Advection in Incompressible Flow," *SIAM J. Numerical Analysis,* vol. 33, pp. 627-665, 1996.
[50] X. Mei, P. Decaudin, B. Hu, and X. Zhang, "Real-Time Marker Level Set on GPU," *Proc. Conf. Cyberworlds,* pp. 209-216, 2008.
[51] E. Mokhberi and P. Faloutsos, "A Particle Level Set Library," technical report, Univ. of Califronia, Los Angeles, 2006.
[52] J.P. Pons, G. Hermosillo, R. Keriven, and O. Faugeras, "Maintaining the Point Correspondence in the Level Set Framework," *J. Computational Physics,* vol. 220, pp. 339-354, 2006.
[53] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *Int'l J. Computer Vision,* vol. 22, pp. 61-79, 1997.
[54] A. Tsai, A. Yezzi Jr, W. Wells, C. Tempany, D. Tucker, A. Fan, W.E. Grimson, and A. Willsky, "A Shape-Based Approach to the Segmentation of Medical Imagery Using Level Sets," *IEEE Trans. Medical Imaging,* vol. 22, no. 2, pp. 137-154, Feb. 2003.
[55] C. Xu and J.L. Prince, "Snakes, Shapes, and Gradient Vector Flow," *IEEE Trans. Image Processing,* vol. 7, no. 3, pp. 359-369, Mar. 1998.
[56] G.K. Rohde, A. Aldroubi, and B.M. Dawant, "The Adaptive Bases Algorithm for Intensity-Based Nonrigid Image Registration," *IEEE Trans. Medical Imaging,* vol. 22, no. 11, pp. 1470-1479, Nov. 2003.
[57] M. Wiebe and B. Houston, "The Tar Monster: Creating a Character with Fluid Simulation," *Proc. ACM SIGGRAPH,* p. 64, 2004.
[58] A. Weiss, D. Hirshberg, and M.J. Black, "Home 3D Body Scans from Noisy Image and Range Data," *Proc. Int'l Conf. Computer Vision (ICCV),* 2011.
[59] I. Baran and J. Popovi, "Automatic Rigging and Animation of 3D Characters," *ACM Trans. Graphics,* vol. 26, article 72, 2007.
[60] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann, "Joint-Dependent Local Deformations for Hand Animation and Object Grasping," *Proc. Graphics Interface '88,* 1988.
[61] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L.J. Guibas, "Meshless Animation of Fracturing Solids," *ACM Trans. Graphics,* vol. 24, pp. 957-964, 2005.
[62] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutre, and M. Gross, "A Unified Lagrangian Approach to Solid-Fluid Animation," *Proc. Eurographics Conf. Point-Based Graphics,* pp. 125-148, 2005.



Blake C. Lucas received the BS degrees in computer science, electrical engineering, and computer engineering from N.C. State University and the MSE degrees in electrical engineering and computer science and the PhD degree in computer science in 2012 both from Johns Hopkins University. He is currently with the JHU Applied Physics Laboratory.

Michael Kazhdan received the PhD degree from Princeton University. He is currently an associate professor in the Computer Science Department at Johns Hopkins University. His more recent research has focused on the challenge of surface reconstruction and considers the manner in which Stokes's Theorem and Laplace's Equation can be used to efficiently and effectively reconstruct high-resolution models from oriented points. Currently, he is working on problems in the domain of image-processing, developing efficient streaming out-of-core algorithms for solving the large sparse linear systems associated with modeling images in the gradient-domain, and on problems in geometry processing, developing hierarchical techniques for efficiently solving the linear systems arising in surface deformation and surface evolution.



Russell H. Taylor (M'76-F'94) received the PhD degree in computer science from Stanford in 1976. After spending 1976 to 1995 as a research staff member and research manager at IBM Research, he moved to Johns Hopkins University, where he is the John C. Malone Professor of computer science with joint appointments in Mechanical Engineering, Radiology, and Surgery and is also the director of the Engineering Research Center for Computer-Integrated Surgical Systems and Technology (CISST ERC). He is the author of more than 300 peer-reviewed publications and book chapters and has received numerous awards and honors. He is a fellow of the IEEE, MICCAI society, etc.