Surface Reconstruction Via Contour Metamorphosis: An Eulerian Approach With Lagrangian Particle Tracking

Ola Nilsson* Linköping University
David Breen† Drexel University
Ken Museth‡ Linköping University

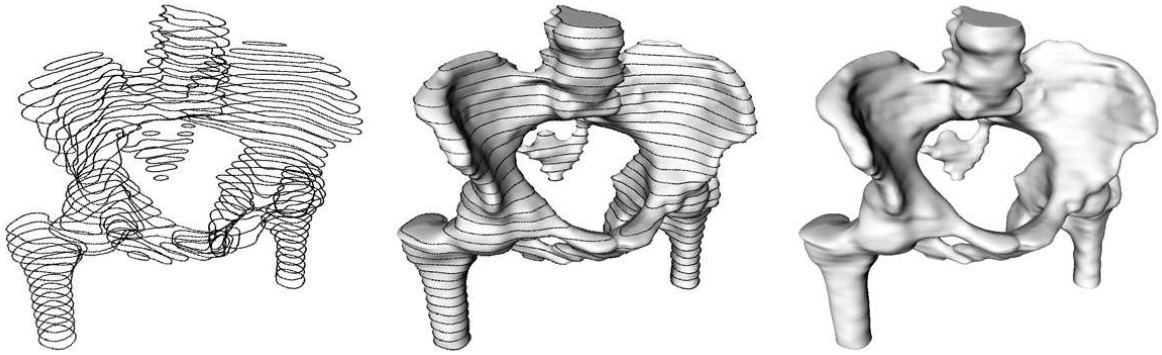

Figure 1: Surface reconstruction with our new method. (left) 35 input contours from a CT scan of the pelvis area bones, resolution 420×300. (middle) Input contours overlaid on the surface reconstruction. (right) The reconstructed surface alone, resolution $420 \times 300 \times 347$.

Abstract

We present a robust method for 3D reconstruction of closed surfaces from sparsely sampled parallel contours. A solution to this problem is especially important for medical segmentation, where manual contouring of 2D imaging scans is still extensively used. Our proposed method is based on a morphing process applied to neighboring contours that sweeps out a 3D surface. Our method is guaranteed to produce closed surfaces that exactly pass through the input contours, regardless of the topology of the reconstruction.

Our general approach consecutively morphs between sets of input contours using an Eulerian formulation (*i.e.* fixed grid) augmented with Lagrangian particles (*i.e.* interface tracking). This is numerically accomplished by propagating the input contours as 2D level sets with carefully constructed continuous speed functions. Specifically this involves particle advection to estimate distances between the contours, monotonicity constrained spline interpolation to compute continuous speed functions without overshooting, and state-of-the-art numerical techniques for solving the level set equations. We demonstrate the robustness of our method on a variety of medical, topographic and synthetic data sets.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Curve, Surface, Solid and Object Representations; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling;

Keywords: 3D reconstruction, contours, level sets

*e-mail: olani@itn.liu.se
†e-mail: david@cs.drexel.edu
‡e-mail: kenmu@itn.liu.se

1 Introduction

A wide variety of objects, animals and specimens are scanned for scientific purposes every day in imaging centers across the globe, producing a steady stream of volumetric datasets. Objects such as developing mouse and frog embryos, rat and monkey brains, nerve cells of all types, bones and even fossils are examined by MRI, CT, ET scanners, as well as physically sliced and imaged to produce these 3D samplings of real objects. Once the objects/specimens have been imaged the resulting volume datasets can be manually segmented. In this process, an experienced anatomist goes over selected slices (*i.e.* images) of the dataset, identifies relevant structures, and circles them with a stylus, producing a series of parallel contours that outline the object of interest.

From these sets of contours it is usually required to produce a high quality, smooth 3D surface model that reconstructs the original object. The reconstructed surface is useful for visualization and further processing, *e.g.* resampling and geometric calculations. An important issue that frequently must be addressed during the reconstruction process is the non-uniform resolution of the scanned datasets. Very often the in-plane (X-Y) resolution of a dataset is greater than the out-of-plane (Z) resolution. This difference can range from a factor of 2 to 10. Many approaches have been proposed that stitch the contours together in order to create a polygon mesh. Another class of solutions takes an implicit approach, where 3D fields are derived by stacking and interpolating 2D distance fields constructed from the individual contours.

Contour stitching algorithms only create polygonal surfaces, thus the resulting reconstructed surfaces have C^0 continuity. Additionally, this class of reconstruction algorithms has not been shown to robustly cope with general, complex, branching structures. We have therefore taken a field-based approach to solving the contour-based reconstruction problem, based on velocity-adjusted contour morphing. With this approach, morphing one contour into the next sweeps out a 3D surface. This is accomplished by equating time in the 2D contour morphing process with the third dimension in the surface reconstruction process. Our approach easily ad-

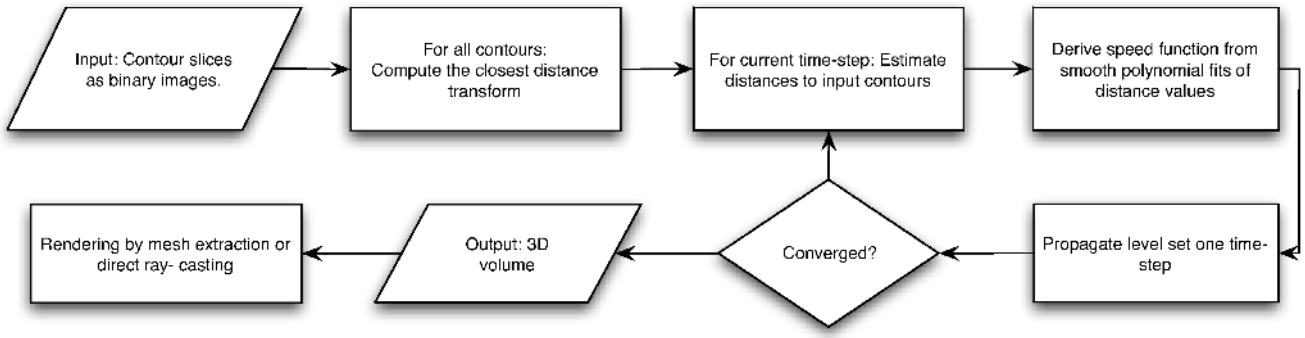

Figure 2: Overview of the reconstruction pipeline described in Section 2.

dresses the *branching problem*, provides a superior technique for interpolating between sparse slices, and produces closed surfaces from contours with both smooth and sharp features. Our work addresses the previously overlooked, but crucial, problem of adjusting the local velocities of the morphing contours in order to guarantee smooth surface transitions at the contour boundaries.

Our approach consists of four major stages. The reconstruction process takes as input a stack of binary images that represents the contours. When completed, a volumetric model is produced, which may be directly rendered or a mesh can be extracted from it for interactive viewing. In the first stage of our approach a 2D signed distance field is computed to each input contour. The contours may also be smoothed before this stage, if desired. Next a 3D surface is produced by performing a series of 2D level set morphs between adjacent contours embedded in the distance fields. This stage is broken up into two steps. First distance estimates are produced that correspond to the arc lengths of trajectories that connect the adjacent contours in the image plane. Next these distances, together with a time-of-arrival, are used to estimate the speeds (in contour normal directions) needed to produce a smooth morph when transitioning between sets of contours. The 3D reconstruction is rendered in the final stage. The complete process is summarized in Fig. 2.

1.1 Previous Work

In the past three decades many significant efforts have addressed the problem of creating surfaces from parallel contours. This work falls into two general categories, contour stitching and field-based methods, which can also be characterized as Lagrangian and Eulerian approaches respectively.

1.1.1 Lagrangian approach: Contour Stitching

The contour stitching approach to surface reconstruction attempts to generate a surface by connecting the vertices of adjacent contours in order to produce a mesh that passes through all contours. These approaches generally need to address the correspondence (how to connect vertices between contours), tiling (how to create meshes from these edges) and branching (how to cope with slices with different numbers of contours) problems.

Keppel [20] and Fuchs et al. [14] described the first algorithms for creating polygonal meshes from a series of contours. The Fuchs work defines the best reconstructed surface as the one with minimal surface area. Many papers have offered incremental improvements to these seminal efforts. Several solutions to the correspondence problem have been proposed, *e.g.* those based on parameterization of the contours [16], contour decomposition [12], Minimum Spanning Trees [25], Angular Bisector Networks [29], medial axes [21] and partial curve matching algorithms [3]. Boissonnat [6] utilizes Delaunay triangulation to cope with branching surfaces. Bajaj et al. [2] provide a unified approach to solving the correspondence, tiling and branching problems by imposing three constraints on the surface when deriving the reconstruction rules. Johnstone et al. [18] describe a method for creating Bezier surfaces from contours with cylindrical topology. Fujimura and Kuo [15] use isotopic deformations to create non-self-intersecting surfaces from nested contours.

1.1.2 Eulerian Approach: Field-Based Methods

Levin [22] presents the seminal field-based approach to surface reconstruction from a series of parallel contours. Given a distance field for each contour, the 2D fields are stacked and interpolated in the z-direction with cubic B-splines. The reconstructed surface is extracted from the resulting 3D field as the zero iso-surface, and in general will only be as smooth as the distance field, *i.e.* C^0. Raya and Udupa [34] extend Levin's approach to time-varying datasets. Jones and Chen [19] suggest that Voronoi diagrams be used to minimize the computation needed for calculating the 2D distance fields. Barrett et al. [5] recursively apply morphological operators (dilation and erosion) to contour images in order to interpolate intermediate gray level values. Cohen-Or et al. [9, 10] introduce the concept, without supporting results, of creating a 3D object from contours by morphing one contour into the next using warp-guided distance field interpolation. Chai et al. [8] present a gradient-controlled partial differential equation method for producing C^1 continuous surfaces from *nested* contours.

1.2 Contributions

We present a novel approach to reconstructing closed 3D surfaces from closed 2D contours. The work described here is the first to demonstrate that smooth 3D models can be created from parallel contours by morphing the contours that sweep out a 3D surface. We propose techniques, based on processing all contours simultaneously, that address the continuity problem at contour boundaries, a problem that is usually caused by connecting only two contours at a time; thus producing smooth surface transitions at the contours.

The approach offers the following additional features:

- **Robustness:** Topology changes occurring between input contours are easily handled. Horizontally overlapping contours are guaranteed to be connected in the reconstruction.

- **Accuracy:** The 3D reconstructions fit accurately to the input contours. Furthermore the input contours are generally not visible in the reconstruction. The 3D surface is at least C^1 continuous in the direction perpendicular to the plane of the contours.
- **Flexibility:** The reconstruction technique allows for any number of input contours (≥ 2), as well as the application of various constraints (*e.g.* derivative and height information, etc.).
- **Efficiency:** The computational complexity of our approach is linear in the size (*i.e.* arc length) of the input contours and thus not dependent on the size of the embedding (*i.e.* images). It should also be emphasized that all of the (level set) computations in our 3D reconstruction method are exclusively performed in 2D employing a fast improved narrow band scheme (see Appendix A).
- **Stability:** We employ proven finite difference schemes for solving Hamilton-Jacobi equations. Additionally we propose an improved equation for level set morphing. Our modified formulation guarantees exact convergence to within the numerical accuracy of the integration scheme.

2 Surface Reconstruction Pipeline

In this section we present the details of the 3D reconstruction pipeline outlined in Fig. 2. The goal of our approach is to generate smooth surfaces that fit to an arbitrary number of parallel contours. When completed it should not be possible to identify the input contours in the final 3D reconstruction, *i.e.* the final surface should be at least C^1 continuous in the direction perpendicular to the contours. However, to allow for sharp features in the input contours, the cross sections of the 3D reconstruction may be C^0 continuous. Since our general approach involves morphing one contour into another we refer to the height dimension perpendicular to the contours as *time*.

2.1 Novel Eulerian Approach: Level Set Model

Explicit curve and surface representations that use vertices or control points can be regarded as Lagrangian approaches since they essentially use body-fixed particles. While this formulation offers many advantages for static geometry, it suffers from significant limitations when representing dynamic geometry: aliasing (*e.g.* undersampling during expansion), failure to easily handle topology changes (*e.g.* merging or bifurcations) and self-intersections (*e.g.* formation of loops or swallowtails). Since our general approach for 3D reconstruction is based on metamorphosis it must cope with complex, changing geometry. A more robust approach to processing dynamic geometric models utilizes an Eulerian formulation, where the deforming interface (contour in 2D or surface in 3D) is implicitly represented as a time-dependent iso-surface or contour of a function discretized on a *fixed* computational grid.

An elegant Eulerian formulation for deforming closed (*i.e.* orientable) interfaces is the level set method [31]. It represents the interface as a time-dependent Euclidian distance function embedded in a Cartesian space of dimension one higher than the interface (*i.e.* co-dimension one). A contour may then be conveniently represented as a 2D image of real numbers that sample the shortest distance function, ϕ, to the contour. A sign convention is used to distinguish between grid points inside (negative) and outside (positive) of the contour. Arbitrary deformation problems may then be recast into a framework that solves the following partial differential equations (PDE),

$$\begin{aligned}\frac{\partial\phi(\boldsymbol{x}(t),t)}{\partial t} &= \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}\cdot\boldsymbol{\nabla}\phi(\boldsymbol{x}(t),t)\\ &= \mathcal{F}(\boldsymbol{x}(t),\phi(\boldsymbol{x}(t),t),\dots)\,|\boldsymbol{\nabla}\phi(\boldsymbol{x}(t),t)|\,,\end{aligned} \tag{1}$$

where $\mathrm{d}\boldsymbol{x}(t)/\mathrm{d}t$ denotes the velocity vector of the deforming interface and $\mathcal{F}()$ is the *speed function* that may depend on a variety of arguments. The geometric interpretation of $\mathcal{F}()$ defines it as the magnitude of the velocity $\mathrm{d}\boldsymbol{x}(t)/\mathrm{d}t$ in the direction normal to the interface at $\boldsymbol{x}$, *i.e.* $\mathcal{F} = \boldsymbol{n}\cdot\mathrm{d}\boldsymbol{x}(t)/\mathrm{d}t$. Also note that the local interface normal, $\boldsymbol{n}$, is given by $\boldsymbol{\nabla}\phi/|\boldsymbol{\nabla}\phi|$ and that the mean curvature is $\boldsymbol{\nabla}\cdot\boldsymbol{n}$, see [26] for details. The level set PDEs, Eq. (1), can be solved efficiently using several numerical techniques, *e.g.* the narrow band schemes [1, 33, 41] and robust finite difference schemes like WENO [23]. In Appendix A we present a fast and yet relatively straightforward narrow band scheme based on an improvement of [33]. This scheme guarantees that our final level set reconstruction algorithm has a computational complexity that is linearly proportional to the size of the input contours, as opposed to the size of the images in which they are embedded. For more general information on methods for solving level set equations we refer the reader to [30, 37].

2.2 Euclidian Distance Fields From Input Contours

The input to our algorithm is a set of contours obtained from parallel 2D slices of a closed surface. We assume that the contours are registered in the same frame of reference and that the individual slices each have an associated height. These heights can be user defined or derived directly from the data sets. For contours from topological maps the third dimension normally corresponds to height values and for medical contours the third dimension may be derived from the distances between the slices.

Since the level set equation, Eq. (1), is a time-dependent Eulerian PDE, it defines an initial value problem. Consequently the first challenge is to derive the initial level sets from the input contours. This amounts to computing Euclidian distance fields from the input contours, which mathematically corresponds to solving the Eikonal equation $|\nabla\phi| = 1$ with associated boundary conditions [37]. This equation can in turn be solved efficiently by a number of numerical methods. For the work presented in this paper we used the Fast Sweeping Method of [43], which is more efficient than the Fast Marching Method of [36, 40]. This stems from the fact that the computational complexity of the former is $O(N)$ in the number of grid points N, as opposed to $O(N\log N)$ for the Fast Marching Method. However we also found the steady-state formulation of [33, 39] to be useful when the input is binary contours, because the time-dependent PDE approach (see Eq. (6)) provides a slight smoothing of the interface, and hence may be used to anti-alias the binary input. We stress, however, that this smoothing is optional.

2.3 Building Block: A Robust Level Set Morphing

The morph of an initial level set model, $\phi(\boldsymbol{x},0) = \phi_{source}(\boldsymbol{x})$, to a final level set, $\phi_{target}(\boldsymbol{x})$, can be formulated and solved with the following PDE,

$$\frac{\partial\phi(\boldsymbol{x}(t),t)}{\partial t} = \left[\phi(\boldsymbol{x}(t),t) - \phi_{target}(\boldsymbol{x})\right]|\boldsymbol{\nabla}\phi(\boldsymbol{x}(t),t)|\,, \tag{2}$$

which is evolved to a steady-state where ϕ and ϕ_{target} are identical. Eq. (2) directs the portions of the initial interface

that are inside the target to expand, and the portions outside to contract. This behavior is produced by the sign convention of ϕ_{target}, and requires that ϕ_{source} and ϕ_{target} overlap; otherwise ϕ_{source} will collapse to a point. Note that the presence of ϕ in the speed function, $\mathcal{F} = \phi - \phi_{target}$, guarantees an exact convergence (*i.e.* steady-state) within the numerical accuracy of the integration scheme. This speed function is numerically more stable than the original formulations of [7, 11] where $\mathcal{F} = -\phi_{target}$, which is unlikely to be exactly zero for the samples in the discretized narrow band. Consequently no true discretized steady-state solution exists, requiring a manual termination of the propagation when the interface is within a grid point's distance to the target. The improved formulation of Eq. (2) will on the other hand converge accurately to the target level set.

It is possible to successively apply a 2D version of Eq. (2) between all pairs of neighboring input contours to create a 3D surface. Time would then correspond to the height coordinate of the 3D surface that sweeps together the contours. While this approach creates a closed surface it does not necessarily produce a desirable result. The 2D morph is not guaranteed to be C^1 continuous in time across contour boundaries. It will be C^0, and in most cases will show major discontinuities in the time derivatives across the input contours. This in turn will lead to 3D reconstructions where input contours are clearly visible, see Fig. 5 (left). To avoid these artifacts a speed function that is at least C^1, or better yet C^2, continuous over time must be defined. See Fig. 5 (right). This is accomplished by assuring that all portions of the contour arrive at the target at the same time, and that the velocity of the morphing contour is the same as it approaches and departs from an input contour.

2.4 Estimating Distances Between Contours: Lagrangian Particle Tracing

Each input contour is assigned a *time-of-arrival*, the time when the morphing contour should reach the input contour. Given our interpretation of time, this value is associated with the height of the input contour. An estimate of the distance traveled by each portion of the deforming contour is required in order to adjust the speed function so that all portions of the contour reach the target simultaneously. This follows from the interpretation of $\mathcal{F}()$ as the speed of a point on a deforming contour in the local normal direction.

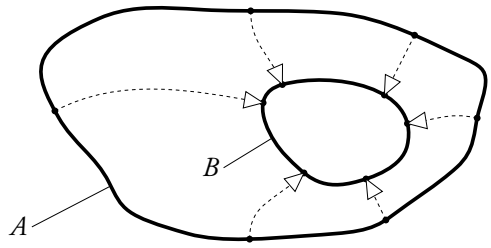

Figure 3: Illustration of the distance estimates between two contours, A and B. Distances are computed as arc-lengths of particle trajectories connecting A and B during a morph defined by Eq. (2).

An effective approach to estimating distances for the speed-function traces particle paths from one contour to the next. The Eulerian morphing of $A \rightarrow B$ can be augmented with Lagrangian particles that keep track of both the traveled distance (*i.e.* the arc-lengths of trajectories between start and end points on the two contours) as well as the point correspondences between A and B.

Tracker particles are first seeded on the zero-crossing of the interface. These particles are advected with an *intermediate* level set using Eq. (2). When the intermediate morph has reached a steady-state, we collect the length of the trajectories traveled by the particles. These distances are then signed (according to the inside/outside convention) and averaged to produce a signed distance estimate for the discrete zero-crossing grid points. A point-to-point correspondence between consecutive contours is also cached making intermediate morphs unnecessary between all contours at all time-steps.

The particle advection for a level set morph from ϕ_A to ϕ_B is implemented by repeating the following steps until a steady-state is reached, *i.e.* $\phi_A = \phi_B$:

1. **Seed** particles randomly on zero-crossing of ϕ_A.
2. **Advect** the particles in the following vector field: $(\phi_B(\boldsymbol{x}) - \phi_A(\boldsymbol{x}))\, \nabla\phi_A(\boldsymbol{x})/|\nabla\phi_A(\boldsymbol{x})|$
3. **Propagate** $\phi_A(\boldsymbol{x})$ with $\mathcal{F} = \phi_B(\boldsymbol{x}) - \phi_A(\boldsymbol{x})$.
4. **Back-project** particles into A using the vector field: $-\phi_A(\boldsymbol{x})\nabla\phi_A(\boldsymbol{x})/|\nabla\phi_A(\boldsymbol{x})|$
5. **Accumulate** the distances traveled to the particles.

Step 1 to 4 are illustrated in Fig. 4. The velocity fields are derived from the geometric interpretation of $\mathcal{F}()$, the fact that the local normal field of ϕ_A is $\nabla\phi_A(\boldsymbol{x})/|\nabla\phi_A(\boldsymbol{x})|$ and Eq. (2). We observe that the back projection step (4) is necessary because discrete integration schemes for solving level set equations have a built-in numerical dispersion [13]. This essentially means Lagrangian particles will almost never follow the level set exactly. Hence the back projection is needed as a correction. The seeding of particles can be adaptive by dynamically adding or deleting particles as the particle densities changes during contour expansion or contraction. However, for the examples presented in this paper a simple over-sampling strategy with 10 initial particles per zero-crossing pixel proved sufficient. Note also that not all particles are guaranteed to reach a target contour. This corresponds to a situation where the particles are seeded on parts of a contour that erode away. It should be emphasized that this is a natural behavior and causes no problems for our subsequent reconstruction. Finally it should be stressed that the above procedure is repeated for each (CFL) time-step which implies that our approximate distance metric will converge to the correct distance as the sweeping level set approaches the input contours.

As a closing remark we note that even though our approach resembles the particle level set method of [13], they are very different. Our method is not designed to modify the level set interface in order to compensate for the numerical dispersion present in the integration scheme. Rather our particles are used for tracking and estimating distances between contours.

2.5 1D Interpolation For The Speed Functions

The approximation of the distances traveled by each contour during morphing is combined with the time-of-arrivals to produce smooth speed functions. Consider a sequence of morphs $A \rightarrow B \rightarrow ... \rightarrow N$ and a particular grid point on the zero-crossing of the current level set. Using the particle tracing technique described above we can estimate the associated signed distances, S_i, that this part of the contour must travel to reach all the (past and future) contours with the time-of-arrivals, t_i, where $i = 1, 2, \ldots, N$. We then fit a smooth polynomial function through these discrete data points and differentiate it to get a speed-function at the considered grid

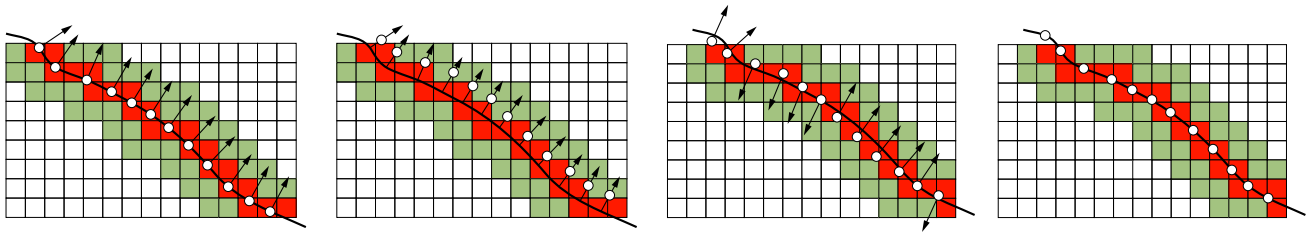

Figure 4: Illustration of the particle advection steps in the narrow band of the level set. (left) Initial configuration with particles seeded on the interface. (middle left) Particles advected in the normal direction. (middle right) Level set advected, and correction vector field used to project particles back onto the interface. (right) Shows the particles projected back on the interface. Note that the zero-crossing pixels are shaded red.

point location. This 1D interpolation is repeated for the remaining zero-crossing grid points on the current level set.

Many differentiable functions can be used to fit the distances and times, using standard curve-fitting techniques. We have investigated several different polynomials, as well as shape-preserving measures. They include linear interpolation, cubic splines and monotonicity constraints.

Figure 5: Reconstruction with linear interpolation (left) and a natural cubic spline (right) for the corresponding speed functions. The inputs are three circular contours, two large ones at the top and bottom and a smaller one at the center.

Simple linear interpolation for the speed-function will create a shape with straight lines connecting the corresponding points on the contours. If instead the aim is to make a smooth shape, a higher order polynomial is needed. One possibility is the natural cubic spline, which is a third order piecewise C^2 polynomial that minimizes strain energy [42]. Two reconstructions from the same input, one using linear interpolation and the other a natural cubic spline, are presented in Fig. 5.

Additional constraints can be applied to the cubic spline in order to control the properties of the reconstruction. If for example the shape is known or desired to be (piecewise) monotonic in time, it is possible to apply a monotonicity filter [17] to the splines. Such a filter imposes the constraint that the spline becomes piecewise monotonic with the potential loss of C^2 continuity. See Fig. 6 for an example. The field of constrained polynomials is vast and several, more complex, methods exist, which may be used to create reconstructions with a variety of shape properties.

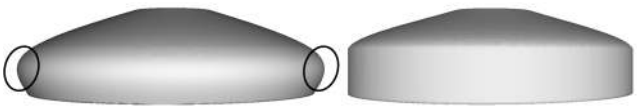

Figure 6: Examples reconstructed with a natural cubic spline that produces the indicated overshooting (left) and monotonicity constraint spline (right). The inputs are three circular contours, a small one at the top and two equally large ones at the center and bottom.

2.6 Velocity Extension and Renormalization

The velocities obtained by interpolation of the particle trajectories are only defined on the zero-crossing grid points. However, the speed-function of Eq. (1) must be defined in the full narrow band of ϕ. Therefore, to extend $\mathcal{F}()$ off of the interface we solve the following transport equation [33, 39]:

$$\frac{\partial \mathcal{F}(\boldsymbol{x},t)}{\partial t} = \mathrm{S}[\phi(\boldsymbol{x},t)]\boldsymbol{\nabla}\mathcal{F}(\boldsymbol{x},t)\cdot\boldsymbol{\nabla}\phi(\boldsymbol{x}(t),t) \tag{3}$$

where

$$\mathrm{S}[\phi(\boldsymbol{x},t)] = \frac{\phi(\boldsymbol{x},t)}{\sqrt{\phi(\boldsymbol{x},t)^2 + |\boldsymbol{\nabla}\phi(\boldsymbol{x}(t),t)|^2}} \tag{4}$$

guarantees that information (*i.e.* the characteristics of Eq. (3)) is propagated in the correct direction off of the interface. $\mathrm{S}[\phi]$ is essentially a smeared sign function of ϕ.

Note that when $\mathcal{F}()$ is defined by velocity extension (Eq. (3)) the corresponding level set propagation is in fact norm conserving, *i.e.* renormalization is not needed to guarantee stability of the numerical scheme. This follows from,

$$\frac{\partial}{\partial t}|\nabla\phi|^2 = 2\boldsymbol{\nabla}\phi\cdot\boldsymbol{\nabla}\frac{\partial}{\partial t}\phi = 2\boldsymbol{\nabla}\phi\cdot(\boldsymbol{\nabla}\mathcal{F}\,|\boldsymbol{\nabla}\phi| + \mathcal{F}\boldsymbol{\nabla}\,|\boldsymbol{\nabla}\phi|) = 0 \tag{5}$$

which makes use of the fact that ϕ is initialized as a Euclidian distance function (*i.e.* $|\boldsymbol{\nabla}\phi| = 1 \Rightarrow \boldsymbol{\nabla}\,|\boldsymbol{\nabla}\phi| = 0$), and $\boldsymbol{\nabla}\phi\cdot\boldsymbol{\nabla}\mathcal{F} = 0$, since Eq. (3) is solved to a steady state.

When using Eq. (2) during advection of the Lagrangian tracker particles, the speed function is derived from closest distance transforms and therefore does not need to be extended. Consequently we must explicitly renormalize ϕ to a Euclidian distance function in order to ensure numerical stability of the morph. For this renormalization we solve

$$\frac{\partial \phi(\boldsymbol{x},t)}{\partial t} = \mathrm{S}[\phi(\boldsymbol{x},t)]\,(|\nabla\phi(\boldsymbol{x},t)| - 1) \tag{6}$$

to a steady state. The sign function in Eq. (6) plays the same role as in Eq. (3).

The third order accurate TVD Runge-Kutta scheme described in [38] is used to accurately integrate these equations with appropriate CFL time steps. Godunov's scheme [35] with a fifth order WENO upwind scheme [23] is used for the numerical discretization in space.

2.7 Closing the ends of the reconstructions

In order to complete the reconstruction the first and last contours must be closed off. One approach is simply to cap the ends with flat planes since no information is available (from the input) beyond the first and last contours. This is done in Figs. 5, 6 and 10. Another approach is to let extrapolation of the calculated speed-functions guide the morph until the surface closes. This approach is used in Fig. 7 and 9. The success of this approach of course depends on the employed interpolation scheme and the fact that the first and last contours are beginning to terminate (*i.e.* are shrinking). The user may manually specify additional external contours

to form a cap. This is done in Fig. 8, where a top-most contour is added and the speed-function is constrained to produce a smooth result. Finally, these three approaches can of course be used in combination.

3 Results

We have applied our reconstruction algorithm to a variety of contour datasets. Fig. 1 presents a reconstruction of the bones of the human pelvis region. It was produced from 35 contours represented by binary images with a resolution of 420×300, and clearly demonstrates our method's ability to produce reconstructions with complex topology. Fig. 7 presents a reconstruction of Mount Everest produced from only five 276×276 binary topographic contour images. This is a good example of how few contours our approach needs to produce useful results. Note also that fine sharp details present in the input contours are correctly captured in the 3D reconstruction. Fig. 8 shows a reconstruction of the upper half of a human figure produced from 12 155×522 binary contour images. Remark that in this example the distances between contours varies significantly, in particular in the facial area. Fig. 9 presents a reconstruction of a mouse embryo produced from only eight 122×187 binary contour images. In this example the first and last contours are rounded simply by extrapolating the level set morphs. Finally Fig. 10 shows an artificial example of a reconstruction from three contour slices, one with two small circles, one with a square and one with a single large circle. Observe how the resulting morph accurately captures the sharp corners of the intermediate square as well as the changing topology.

All results presented in this paper were rendered using the standard mesh extraction technique of [24]. The computational times on a 2.5GHz Macintosh G5 were 107 CPU-seconds for Mt. Everest, 220 CPU-seconds for the mouse embryo, 240 CPU-seconds for the human torso and 1100 CPU-seconds for the pelvis data set. It should also be stressed that none of the reconstructions required any user input other than the initial contours with associated time-of-arrivals.

Figure 10: An artificial reconstruction with changing topology. Note that our method accurately captures the difficult rectangular shape. The interpolation scheme is using a natural cubic spline.

4 Conclusions and Future Work

We have presented a robust method for 3D reconstruction of closed surfaces from sparsely sampled parallel contours. Our method is based on a morphing process applied to neighboring contours that sweeps out a 3D surface as one contour morphs into the next. The morph is performed with an Eulerian formulation (*i.e.* fixed grid) augmented with Lagrangian particles (*i.e.* interface tracking). This is accomplished by propagating the input contours as 2D level sets with carefully constructed continuous speed functions. We utilize particle advection to estimate distances between the contours, monotonicity constrained spline interpolation to compute continuous speed functions without overshooting, and state-of-the-art numerical techniques for solving the level set equations. Our approach robustly reconstructs objects with complex branching structures, provides a superior technique for interpolating between sparse slices, and produces closed surfaces from contours with both smooth and sharp features. It addresses the previously overlooked, but crucial, problem of adjusting the local velocities of the morphing contours in order to guarantee smooth surface transitions at the contour boundaries.

Future work includes implementing user interaction techniques for processing datasets with non-overlapping contours, similar to [10]. This will allow the user to control the direction of the morph, thus offering an approach to applying expert knowledge about anatomically correct relationships between different segments of the reconstructed object. As described in [27] multiple non-aligned datasets may be generated from a single scanning session for a particular specimen. Our approach may be extended to create 3D surfaces from the contours of these non-uniform, arbitrarily-oriented, multiple datasets. Finally we plan to extend our level set method with the more efficient data structures and algorithms of [28] which will allow for reconstruction at extreme resolutions.

5 Acknowledgments

This work was partially funded by Swedish Science Council grant VR-621-2004-5017 and National Science Foundation grant ACI-0083287. The pelvis dataset was provided by Gill Barequet of the Technion, and the mouse embryo dataset was provided by the Caltech Biological Imaging Center. The Mt. Everest dataset is courtesy of Kai Hormann.

A A Fast Narrow Band Implementation

For optimal computational complexity we use a modified version of the narrow band scheme presented in [33]. It employs two dynamic tubes that enclose the level set interface; a T tube of width γ and an N tube that is one pixel wider than the T tube. We employ simple C-style arrays as defined in the following pseudo-code to implement efficient data structures for these tubes.

```
int dim = 1,X[m],Y[m],Mask[m][m];
foreach pixel (i, j) do
    Mask[i][j] = 0; /*          outside both tubes    */
    if |φ(i, j)| < γ then
        Mask[i][j] = 2; /*       inside both tubes    */
        X[dim] = i; Y[dim++] = j;
    else if |φ(i ± 1, j ± 1)| < γ then
        Mask[i][j] = 1; /*       inside the N tube    */
        X[dim] = i; Y[dim++] = j;
    end
end
```

m is always chosen to be larger then the number of pixels in the narrow band (dim). The level set equation is then solved exclusively in the T tube by looping over all elements in the arrays, for $k = 1, \ldots, dim$, and only updating elements for which $Mask[X[k]][Y[k]] = 2$. Next, renormalization is performed by solving Eq. (6) in the N tube, *i.e.* for pixels where $Mask[X[k]][Y[k]] \geq 1$. This implies that the overall computational complexity of solving the level set equation is *linear* in the size of the interface and not of the embedding. To rebuild N and T after each time propagation we could apply the above algorithm again (as suggested in [33]), but this is inefficient since it visits all pixels. To maintain a linear computational complexity we instead use the following algorithm.

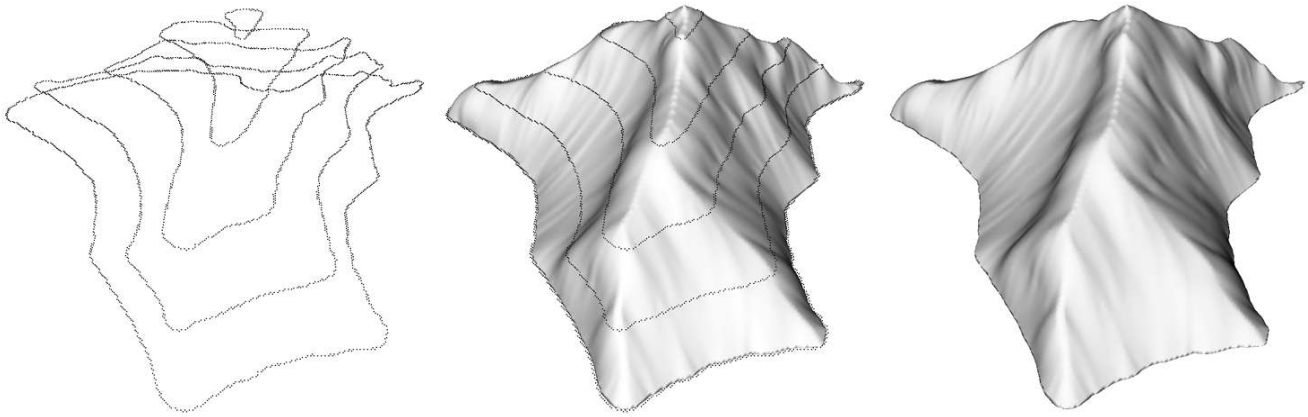

Figure 7: Reconstructed model of Mt Everest from only five topographic contours. The interpolation scheme for the speed-function is a natural cubic spline and the top of the reconstruction is closed with speed function extrapolation. The final resolution is $276 \times 276 \times 97$.

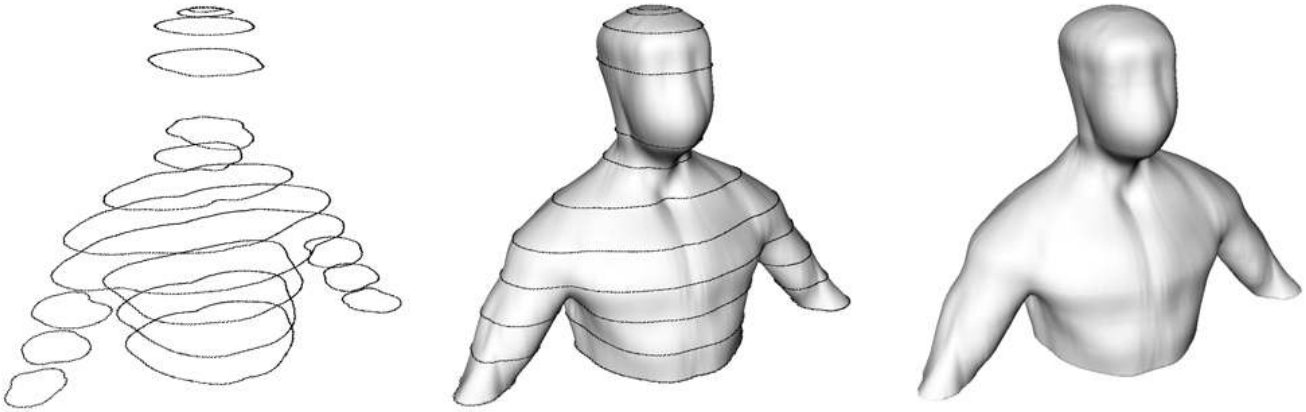

Figure 8: Reconstructed human model from 12 input contours. Note that the method nicely sweeps out the face even the though the input is very sparse. The resolution is $155 \times 522 \times 270$. The interpolation scheme for the speed-function is a natural cubic spline and the top of the reconstruction is closed by manually adding an extra contour combined with spline extrapolation.

foreach *pixel* $(i,j) \in N_{old}$ **do**
  if $|\phi(i,j)| < \gamma$ **then**
    add (i,j) to T_{new} and N_{new};
  else
    foreach (i,j)*'s neighbors* $(p,q) \notin N_{new}$ **do**
      if $|\phi(p,q)| < \gamma$ **then** add (i,j) to N_{new};
    end
  end
  if $(i,j) \notin T_{old}$ *but* $(i,j) \in T_{new}$ **then**
    foreach (i,j)*'s neighbors* $(p,q) \notin N_{new}$ **do**
      add (p,q) to N_{new}
    end
  end
end

References

[1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118(2):269–277, 1995.

[2] C.L. Bajaj, E.J. Coyle, and K-N. Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing*, 58:524–543, 1996.

[3] G. Barequet, D. Shapiro, and A. Tal. Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices. *The Visual Compouter*, 16(2):116–133, 2000.

[4] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63:251–272, 1996.

[5] W. Barrett, E. Mortensen, and D. Taylor. An image space algorithm for morphological contour interpolation. In *Proc. Graphics Interface*, pages 16–24, 1994.

[6] J.-D. Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, 1988.

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

[8] J. Chai, T. Miyoshi, and E. Nakamae. Contour interpolation and surface reconstruction of smooth terrain models. In *Proc. IEEE Visualization*, pages 27–33, 1998.

[9] D. Cohen-Or and D. Levin. Guided multi-dimensional reconstruction from cross-sections. In F. Fontanella, K. Jetter, and P.-J. Laurent, editors, *Advanced Topics in Multivariate Approximation*, pages 1–9. World Scientific Publishing Co., 1996.

[10] D. Cohen-Or, D. Levin, and A. Solomovici. Contour blending using warp-guided distance field interpolation. In *Proc. IEEE Visualization*, pages 165–172, 1996.

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

[12] A.B. Ekoule, F.C. Peyrin, and C.L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, 10(2):182–199, 1991.

[13] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1):83–116, 2002.

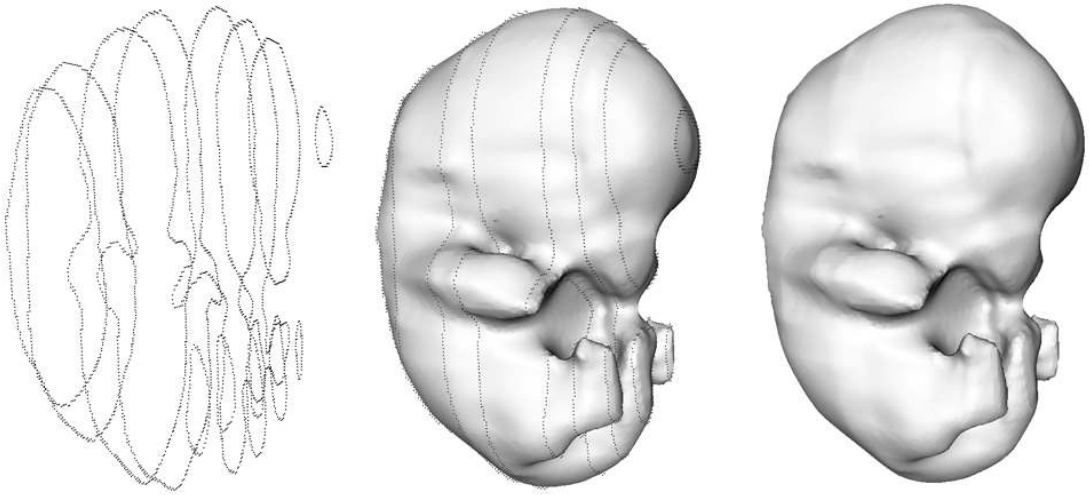

Figure 9: Mouse embryo from an MRI scan reconstructed from only eight input contours. The final resolution is $122 \times 187 \times 98$. The interpolation scheme for the speed-function is a natural cubic spline and the ends of the reconstruction are closed with speed function extrapolation.

[14] H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[15] K. Fujimura and E. Kuo. Shape reconstruction from contours using isotopic deformation. *Graphical Models and Image Processing*, 61:127–147, 1996.

[16] S. Ganapathy and T.G. Dennehy. A new general triangulation method for planar contours. In *Proc. SIGGRAPH '78*, pages 69–75, 1978.

[17] J. M. Hyman. Accurate monotonicity preserving cubic interpolation. *SIAM Journal of Scientific and Statistical Computing*, 4(4):645–654, 1983.

[18] J. Johnstone and K.R. Sloan. Tensor product surfaces guided by minimal surface area triangulations. In *Proc. IEEE Visualization*, pages 254–261, 1995.

[19] M. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):75–84, 1994.

[20] E. Keppel. Approximating complex surface by triangulation of contour lines. *IBM Journal of Research and Development*, 19:2–11, 1975.

[21] R. Klein, A.G. Schilling, and W. Strasser. Reconstruction and simplification of surfaces from contours. *Graphical Models*, 62:429–443, 2000.

[22] D. Levin. Multidimensional reconstruction by set-valued approximation. *IMA Journal of Numerical Analysis*, 6:173–184, 1986.

[23] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially nonoscillatory schemes. *J. Comput. Phys.*, 115:200–212, 1994.

[24] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construct ion algorithm. In *Proc. SIGGRAPH '87*, volume 21, pages 163–169, 1987.

[25] D. Meyers, S. Skinner, and K. Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, 1992.

[26] K. Museth, D. Breen, R. Whitaker, S. Mauch, and D. Johnson. Algorithms for interactive editing of level set models. *Computer Graphics Forum*, accepted for publication, 2005.

[27] K. Museth, D. Breen, L. Zhukov, and R. Whitaker. Level set segmentation from multiple non-uniform volume datasets. In *Proc. IEEE Visualization 2002*, pages 179–186, October 2002.

[28] M. Nielsen and K. Museth. Dynamic Tubular Grid: An efficient data structure and algorithms for high resolution level sets. *Linköping Electronic Articles in Computer and Information Science*, 9(001):ISSN 1401–9841, 2004. Accepted for publication in *Journal of Scientific Computing* January 26, 2005.

[29] J. M. Oliva, M. Perrin, and S. Coquillart. 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized voronoi diagram. *Computer Graphics Forum*, 15(3):397–408, 1996.

[30] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, Berlin, 2002.

[31] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[32] S. Osher and C.-W. Shu. High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Num. Anal.*, 28:907–922, 1991.

[33] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155(2):410–438, 1999. Pseudo code on page 418 has errors - see instead our appendix A.

[34] S.P. Raya and J.K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9(1):32–42, 1990.

[35] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM J. Num. Anal.*, 29:867–884, 1992.

[36] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. of the National Academy of Sciences of the USA*, 93(4):1591–1595, February 1996.

[37] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, second edition, 1999.

[38] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[39] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114(1):146–159, 1994.

[40] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. Automat. Contr.*, 40:1528–1538, September 1995.

[41] R. Whitaker. A level-set approach to 3D reconstruction from range data. *Int. J. Comput. Vision*, 29(3):203–231, 1998.

[42] G. Wolberg and I. Alfy. Monotonic cubic spline interpolation. In *CGI '99: Proc. International Conference on Computer Graphics*, page 188, 1999.

[43] H.K. Zhao. Fast sweeping method for eikonal equations. *Mathematics of Computation*, 74:603–627, 2004.